**US Army Corps
of Engineers**
Construction Engineering
Research Laboratories

# The Support Environment for Design and Review (SEDAR) for Flat and Low-Slope Roofs

by
Michael C. Fu
E. William East
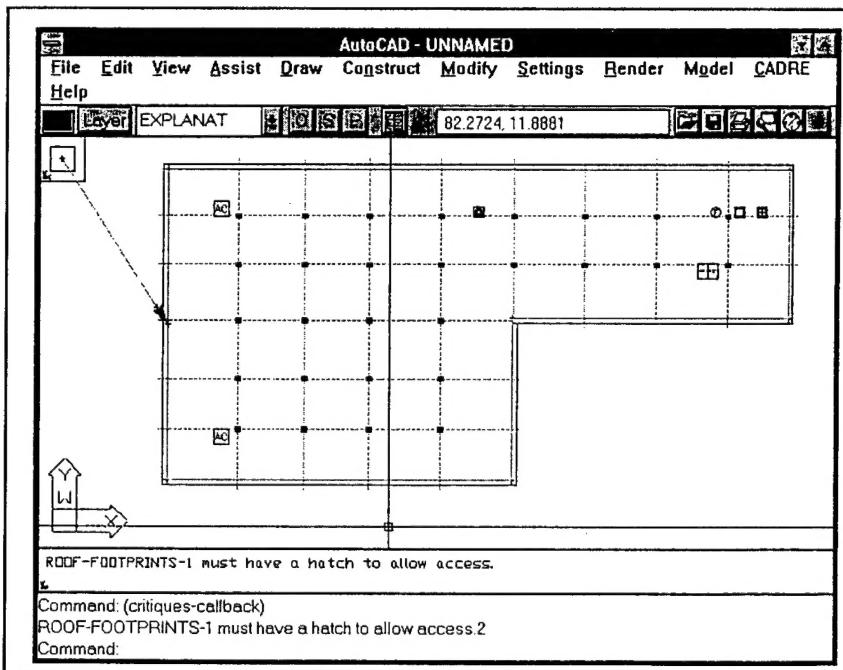


1996118 017

This report describes an expert critiquing system, the Support Environment for Design And Review (SEDAR), that uses a task-based model of design for flexible control of its multi-strategy critiquing abilities. SEDAR has been developed for the flat and low-slope roofing domain, a subfield of the building design domain. It is designed to support the existing design/review protocol for roof design for the U.S. Army Corps of Engineers.

SEDAR offers three critiquing strategies. The incremental *error prevention* strategy is intended to help users avoid errors by visually displaying "off-limits" areas before errors can be made. The incremental *error correction* strategy's intent is to give immediate feedback to the user during the design process, so that the errors may be corrected before their effects are propagated to subsequent parts of the design. The batch-processing *design review* strategy is intended to allow the user to conduct reviews on a design after particular roof subsystems are completed, as some errors can not be detected until this stage.

DTIC QUALITY INSPECTED 1

## USER EVALUATION OF REPORT

REFERENCE: USACERL Technical Manuscript 96/99, *The Support Environment for Design and Review (SEDAR) for Flat and Low-Slope Roofs*

Please take a few minutes to answer the questions below, tear out this sheet, and return it to USACERL. As user of this report, your customer comments will provide USACERL with information essential for improving future reports.

1.   Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which report will be used.)

_____

_____

_____

2.   How, specifically, is the report being used? (Information source, design data or procedure, management procedure, source of ideas, etc.)

_____

_____

3.   Has the information in this report led to any quantitative savings as far as manhours/contract dollars saved, operating costs avoided, efficiencies achieved, etc.? If so, please elaborate.

_____

_____

4.   What is your evaluation of this report in the following areas?

   a. Presentation: _____

   b. Completeness: _____

   c. Easy to Understand: _____

   d. Easy to Implement: _____

   e. Adequate Reference Material: _____

   f. Relates to Area of Interest: _____

   g. Did the report meet your expectations? _____

   h. Does the report raise unanswered questions? _____

i. General Comments. (Indicate what you think should be changed to make this report and future reports of this type more responsive to your needs, more usable, improve readability, etc.)

_____

_____

_____

_____

_____

_____

5. If you would like to be contacted by the personnel who prepared this report to raise specific questions or discuss the topic, please fill in the following information.

Name: _____

Telephone Number: _____

Organization Address: _____

_____

_____

6. Please mail the completed form to:

Department of the Army
CONSTRUCTION ENGINEERING RESEARCH LABORATORIES
ATTN: CECER-TR-I
P.O. Box 9005
Champaign, IL 61826-9005

# REPORT DOCUMENTATION PAGE

**Form Approved**
**OMB No. 0704-0188**

| 1. AGENCY USE ONLY *(Leave Blank)* | 2. REPORT DATE<br>September 1996 | 3. REPORT TYPE AND DATES COVERED<br>Final |
|---|---|---|

**4. TITLE AND SUBTITLE**
The Support Environment for Design and Review (SEDAR) for Flat and Low-Slope Roofs

**5. FUNDING NUMBERS**
4A162784
AT41
AP6

**6. AUTHOR(S)**
Michael C. Fu and E. William East

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

U.S. Army Construction Engineering Research Laboratories (USACERL)
P.O. Box 9005
Champaign, IL 61826-9005

**8. PERFORMING ORGANIZATION REPORT NUMBER**

TM 96/99

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
Headquarters, U.S. Army Corps of Engineers (HQUSACE)
ATTN: CEMP-CE
20 Massachusetts Avenue, NW.
Washington, DC 20314-1000

**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES**
Copies are available from the National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161.

**12a. DISTRIBUTION / AVAILABILITY STATEMENT**

Approved for public release; distribution is unlimited.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT (Maximum 200 words)**

This report describes an expert critiquing system, the Support Environment for Design And Review (SEDAR), that uses a task-based model of design for flexible control of its multi-strategy critiquing abilities. SEDAR has been developed for the flat and low-slope roofing domain, a subfield of the building design domain. It is designed to support the existing design/ review protocol for roof design for the U.S. Army Corps of Engineers.

SEDAR offers three critiquing strategies. The incremental error prevention strategy is intended to help users avoid errors by visually displaying "off-limits" areas before errors can be made. The incremental error correction strategy's intent is to give immediate feedback to the user during the design process, so that the errors may be corrected before their effects are propagated to subsequent parts of the design. The batch-processing design review strategy is intended to allow the user to conduct reviews on a design after particular roof subsystems are completed, as some errors can not be detected until this stage.

**14. SUBJECT TERMS**
roofing
expert systems
design evaluation

**15. NUMBER OF PAGES**
132

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT<br>Unclassified | 18. SECURITY CLASSIFICATION OF THIS PAGE<br>Unclassified | 19. SECURITY CLASSIFICATION OF ABSTRACT<br>Unclassified | 20. LIMITATION OF ABSTRACT<br>SAR |
|---|---|---|---|

# Foreword

# Contents

# List of Figures

# 1  Introduction

This report describes an expert critiquing system, the Support Environment for Design And Review (SEDAR) [Fu 1994], that has been developed for the flat and low-slope roofing domain, a subfield of the building design domain. The primary contribution of SEDAR is its use of a task-based model of design, called the *Designer's Task Model* (DTM), to guide its critiquing actions. SEDAR's use of the DTM allows it to flexibly track individual roof designers' problem solving strategies, and to provide the most relevant critiques at each step in the design process.

SEDAR was developed with two primary objectives. The first objective was to investigate theories concerning expert critiquing systems and the role of computers as collaborators in the design process. The development of the DTM was an attempt to address certain shortcomings of existing expert critiquing systems. The second objective was to create a system that solved a significant problem in the Architect/ Engineer/Contractor (A/E/C) community, specifically to facilitate the processing of uncompleted design reviews delaying building development and construction. In particular, SEDAR is a resource or tool that a designer may use to apply the collected expertise of other designers and reviewers efficiently to the current design task. To achieve this function, SEDAR supports and improves upon the existing design/review process of A/E/C firms. The design/review process, described in more detail later in this chapter, is essentially one of iterative redesign.

Human critics give reasoned opinions based on observations of their subjects' problem-solving behavior. They help their subject recognize and correct misconceptions, biases, and poor habits. They may also help to fill in gaps in the subject*s knowledge of the task domain. The goal of expert critiquing systems is to augment users' problem-solving abilities in the system's domain of expertise in a similar fashion to that of human critics. The computer critic interacts with the human user in a fashion similar to Figure 1.

Such systems are one of the many types of decision support technologies that have evolved over the past two decades in the field of artificial intelligence. They have been constructed for various task domains, including medical diagnosis [Miller 1986], circuit design [Spickelmier 1988], kitchen layout [Fischer 1991], and antenna layout on ships [Zhou 1989].
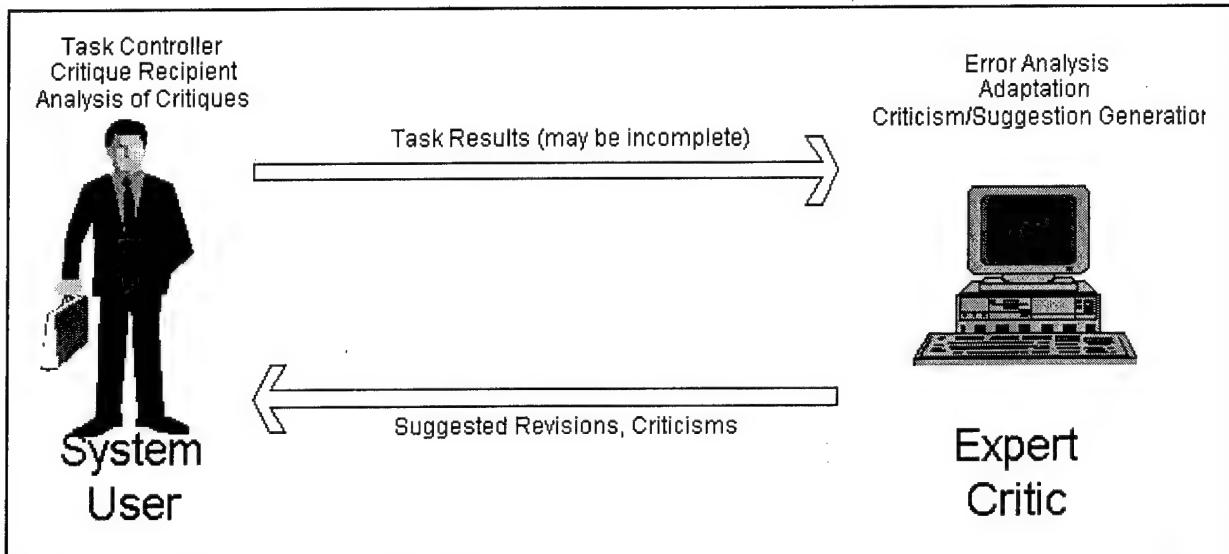
**Figure 1. The Critiquing Interaction Model.**

A significant problem with many existing critiquing systems is their inability to adapt their critiques to the changing goals and beliefs of the user. As the needs of the user evolve over the design process, the critiquing system must be able to change the content of its critiques to present the most focused, salient advice with respect to the existing design and the user's design goals. Another problem with expert critiquing systems is their inability or unwillingness to yield control of the problem solving process to the human. By forcing the human along inflexible solution paths, critiquing systems may inadvertently annoy rather than help the user.

SEDAR's primary contribution to the field of expert critiquing systems is its use of a DTM to direct the expert critiquing system in a focused but flexible way. Developed from roofing literature and interviews with domain experts, the DTM is a hierarchically organized model of an experienced roof designer's task structure for roof design. The DTM is used to map observed user actions onto beliefs about the user's intentions. These beliefs then influence the response of the system to the user's actions. Unlike the user goal models of current expert critiquing systems [Mastaglio 1990; Fischer 1993], the DTM is a process-based representation used in a way that permits it to adapt to the problem-solving behavior of individual users.

Another contribution of this work is an assessment of the utility of different critiquing strategies in the context of supporting experienced designers in the roofing domain. The basic definition of expert critiquing systems presented above encompasses an enormous variety of advice-giving strategies. A system may give short, one-shot critiques to jog the memory of the user or it may give longer, more involved explanations. The criticism may be preventative or correctional (or both). A system

may be aggressive, actively reasoning about the design and notifying the user of its opinions when necessary, or it may be passive, allowing the user to determine when the critiquing process should take place.

SEDAR supports three critiquing strategies. The *error prevention* strategy seeks to avoid errors on the roof design by cuing the user before an error can occur. The *error correction* strategy tests for existing errors on the design. The *design review* strategy provides a tool that a designer may use to produce critiques on particular roof subsystems. Testers of the system used each of the critiquing strategies and were asked to provide an assessment of each.

Just as many different forms of advice may be generated by a critiquing system, there are also many ways of expressing that advice to the user. Early expert systems like ONCOCIN and ATTENDING were mainly text-based; more recent critiquing systems like JANUS, CLEER, and SEDAR have begun to explore how graphical/pictorial representations may be used to improve the effectiveness of the critiquing process. SEDAR's use of integrated graphical/textual critiques displayed directly on the roof design is another contribution of this work.

The remainder of this chapter introduces the roof design task, discusses how the design/review protocol of A/E/C firms supports this task, and presents an overview of this research effort.

## The Roof Design Task

The low-slope roofing domain was chosen for use in SEDAR for a number of reasons. The volume of built-up roofing (a type of flat/low-slope roofing) alone annually installed in the United States totals about 3 billion square feet [Griffin 1982], enough to cover Washington, DC twice. A conservative estimate by the National Bureau of Standards is that 4 to 5 percent of these roofs fail prematurely, causing a huge economic burden on the buildings' owners for repair and replacement, a significant legal threat to architects, contractors, and manufacturers involved in the roofing industry, and in a few extreme cases physical danger to those living or working within the affected building.

The domain itself is complex in that the roof designer must juggle a multitude of economic, technical, political, and even social factors while creating a roof design. Economically and politically speaking, the building's roof design lags far behind the more "glamorous" building subsystems competing for the building owner's money and the architect's time and interest. Technically, a number of factors make modern

roof design increasingly complex. The proliferation of new materials, expanding roof dimensions, more rigorous roof-performance requirements, and modern trends in flexible building design all combine to make the roof designer's task more difficult.

However, the roof design domain is not so complex as to be intractable. As a subsystem of the building system, roof design involves only a highly restricted subset of both the tasks and objects relevant to building design. Furthermore, because of the recognized importance of the roofing system, handbooks defining a clear standard of quality roof design are readily available.

The responsibility of creating a high quality, durable roof begins with the designer. Although problems in the roof may be caused by poor roof application techniques, cost-cutting attempts, and even ignorance, a clear, complete, and correct roof plan will help to eliminate many of the problems experienced in the construction and maintenance phases of the roof's life cycle.

Studies of roof designers showed that they tend to decompose the overall design task into subtasks focused on designing subsystems of the roof system. However, roof design is not truly hierarchically decomposable; each subtask cannot be considered in isolation from the other subtasks. Because of the many interdependencies among subtasks, roof design is only *heterarchically decomposable* [Case 1994]. The knowledge gained from these studies led to the proposal and use of the DTM to represent this task decomposition and to provide flexible support for the roof designer.

## The Design/Review Process

The Design/Review Process is used by the A/E/C community to help ensure the quality of designs. The process is one of iterative redesign, similar to the method described by Brown [1986]. It is also a significant contributor to the length of the (often) months-long design/review process.

Throughout the building design process, two groups of people work on the building design. The design team is responsible for the synthesis of design, from developing functional specifications for the building, through interactions with the customers, to finalizing the set of design documents for the construction stage. At many points in the design process, the design team submits the existing design documents to a review team, who periodically check the documents for inconsistencies, errors, and other suboptimal aspects of the design. Figure 2 displays the relationship between the design and review teams.
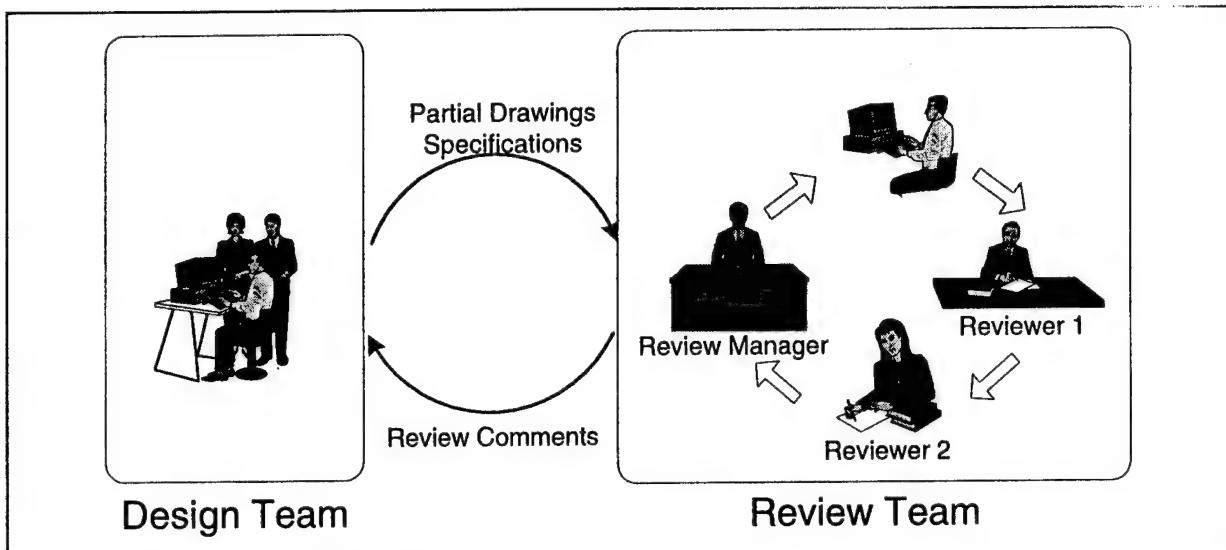
Figure 2. The design/review process.

Each member of the review team is an expert in a different engineering field; for example, in a building design review team, there may be an architect, electrical engineer, structural engineer, mechanical engineer, etc. The design documents pass from member to member of the review team, as each critiques the design according to his/her area of expertise. The critiques are accumulated as a set of review comments and are returned to the design team along with the design documents. Reviews must be conducted at least twice during the design process. The design process itself may be divided into four stages of design completion: conceptual design, intermediate design, detailed design, and the final design (Figure 3).

During the conceptual design stage, the design team works closely with the customers to determine the functional requirements of the building. Functional requirements vary from building to building. For example, a building with an indoor pool requires additional ventilation capabilities to prevent excessive moisture from building up within the room, and a computer laboratory requires a larger than normal air conditioning capacity, which may translate into larger air conditioning units and ventilation ducts. After clarifying the customer's functional requirements, some rough, high-level design decisions
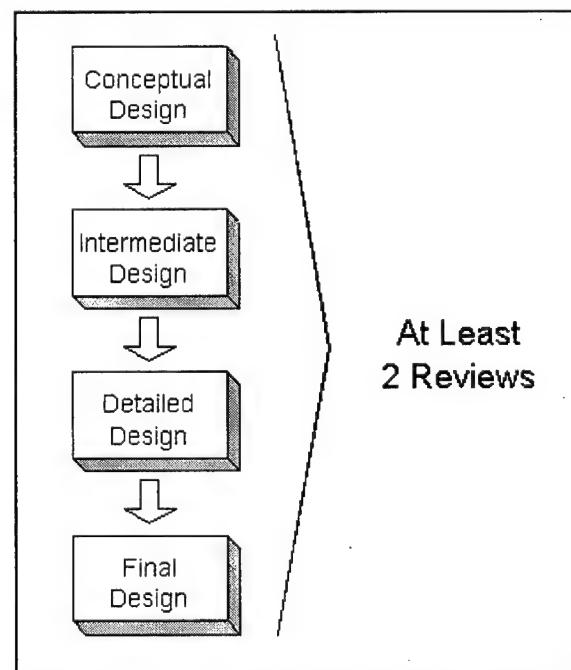


Figure 3. Design completion stages and reviews.

are made about the roof, including the layout of major roof components. During the intermediate design stage, the design team adds more detail to the building drawing. For example, large pieces of mechanical equipment, such as air conditioning units, roof access mechanisms, and walkways that allow access to the equipment on the roof, are laid out on the design. During the detailed design stage, further detail is added to the design; appropriate to this stage would be equipment and roof flashing details. Also, a set of specification documents are created, which describe the composition of the building components and contain further design and construction guidelines. In the final design stage, the building drawings and specification documents are completed.

Based on conversations with review personnel during visits to two USACE division offices, where reviews are carried out for building designs for each office's geographic region, three conclusions were made:

1.  Because of the nature of the design/review protocol, the process of designing a building is an inherently time-consuming, resource-intensive process. In particular, the assignment of qualified review personnel to other tasks in division offices created resource bottlenecks among the existing reviewers—a single reviewer would have dozens of backlogged design documents to examine. Furthermore, the design checking process is slow and tedious; the reviewer must methodically check for all possible errors.

2.  Due to the time-constrained environment, many reviewers are often forced to "cut corners" with respect to reviews of design documents. For example, a reviewer might look at a few critical aspects of the design to develop a feel for the competence of the design team. If they meet the reviewer's criteria, then the reviewer is apt to check the remainder of the design less carefully, because he or she feels that the design team is less likely to make errors. While this viewpoint is justifiable given the time limitations imposed by the existing work environment, a rigorous, complete design check is certainly preferable, if it can be accomplished efficiently.

3.  Designs are reviewed only at certain predefined points in the design process. Again, this is an artifact of the existing design/review protocol. Given the amount of time that a design team has to wait while the review team checks the design documents, the current protocol of performing reviews once or twice during the design process is an effort to balance design generation with error discovery and correction. If documents were reviewed more often, errors would be caught earlier in the design process, and less redesign would be needed to correct those errors. However, this carries the additional cost of holding more

reviews. If documents were reviewed less often, the overall time cost of performing reviews would decrease, but the possibility of extensive redesign due to an earlier error would increase.

## Project Overview

SEDAR's answer to the problems of the existing design/review process is to better integrate the design and review phases of the protocol. First, the type of design rule checking conducted by reviewers should be performed completely, efficiently, and continuously during the design process. The benefit of continual checking is that a designer is notified of design flaws immediately after the error has been made. Besides alerting the designer when errors have been made, the opportunity exists to prevent the errors in the first place. The goal of expert critiquing systems is not to replace the human in either the design or review processes, but rather to augment his/her abilities with a flexible, intelligent tool.

The incremental critiquing strategies of SEDAR (error prevention and error correction) are best suited to assist the designer. The designer may also conduct automated reviews on the existing design by invoking the design review critic. Thus one use for SEDAR is as a surrogate member of the design team, applying knowledge from constructibility reviews to assist the roof designer (Figure 4). Use of SEDAR in this environment is the primary focus of this work.

SEDAR can also provide assistance to reviewers. The design review critic helps a roof design reviewer to thoroughly check published design codes [NRCA 1985], which establish the minimum requirements of a roof design. The reviewer may then check the roof on higher-level issues (e.g., optimality of roof-mounted equipment, walkway, and drainage system design) that are currently beyond SEDAR's capabilities. A depiction of this scenario is shown in Figure 5.

SEDAR may be characterized as an integrated design environment that uses the critiquing paradigm to guide the content and timing of its user assistance. Central to the critiquing paradigm of SEDAR is the use of the DTM



Constructibility Review Knowledge Applied to Design

SEDAR

Roof Designer (Architect)
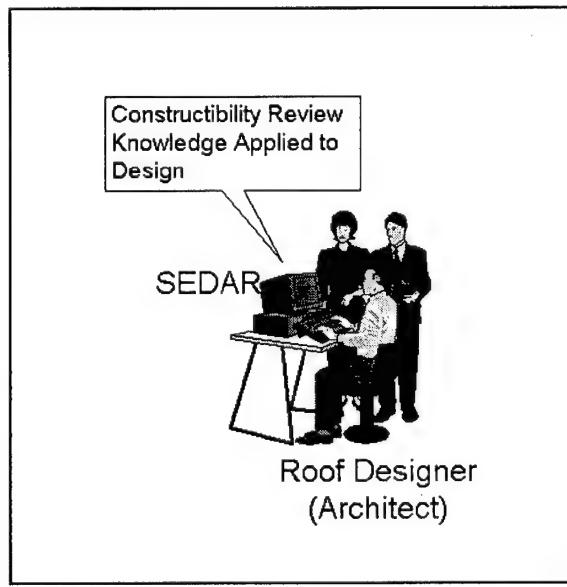
Figure 4. Use of SEDAR in the design team.
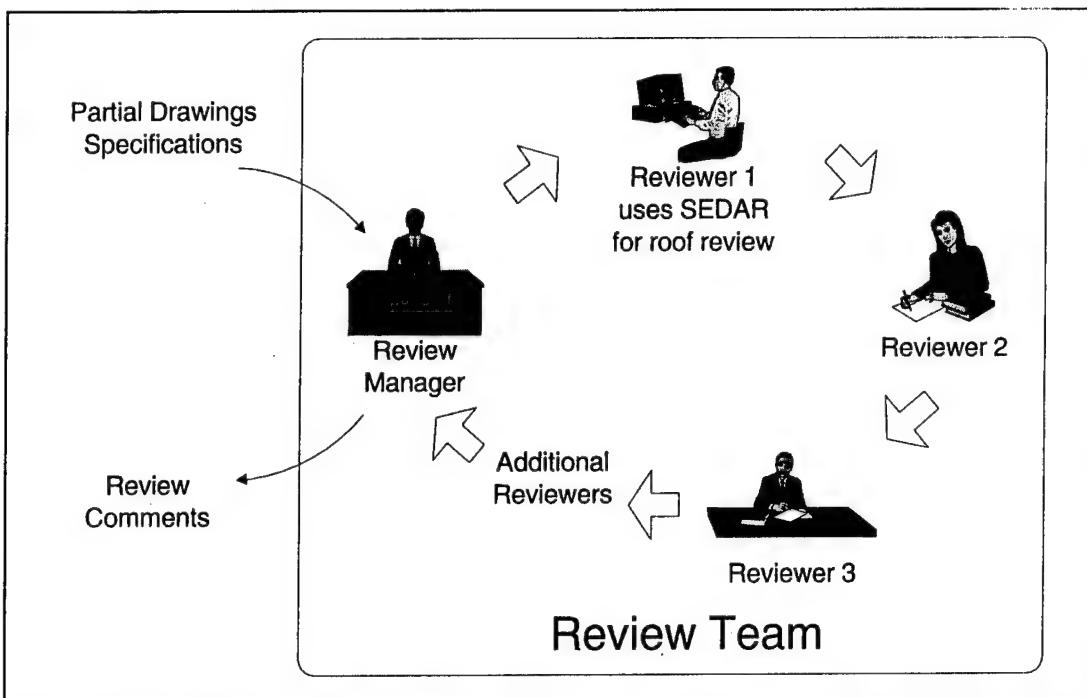
**Figure 5. Use of SEDAR in the review team.**

to control the critiquing process, something unique in the domain of expert critiquing systems (Chapter 4).

The evaluation of the SEDAR research prototype yielded interesting results. Roof designers liked the system in general but had strong responses to particular system elements. They also provided considerable insight on how to improve the system to meet their needs.

## Report Outline

Before starting an in-depth discussion of the research for this project, some background information will be presented. Chapter 2 contains a discussion of several issues along which expert critiquing systems may be characterized. These various dimensions define the purpose, strategies, and flexibility of critiquing systems. Past work on critiquing systems, as well as related noncritiquing design systems are summarized according to these issues in Chapter 3. A high-level description of SEDAR made in terms of the issues is also included. The research contributions, summarized above, are discussed in more detail in Chapter 4. The background information concludes with a short overview of the low-slope-roofing domain in Chapter 5. Besides diagraming the components of a basic flat/low-slope roof, a closer look is taken at why roofs fail and at how human designers approach the roof design task. Chapter 6 is a component-by-component overview of the SEDAR architecture.

Its purpose is to provide a concise functional description of the capabilities of SEDAR. Chapters 7 and 8 look at SEDAR from different perspectives. Chapter 7 describes SEDAR's knowledge representation, while the operation of SEDAR is illustrated using a series of examples in Chapter 8. Chapter 9 describes the evaluation/testing strategy that was used and the results of the experiments. The design and implementation of SEDAR has raised many interesting issues, which range from rethinking the philosophy of SEDAR to dealing with domain-specific issues. Some of the most important issues are presented in Chapter 10. Finally, Chapter 11 offers some conclusions made at this stage of the research effort.

# 2  Issues in Expert Critiquing Systems

This chapter provides an overview of many of the important issues in expert critiquing systems, which define dimensions along which expert critiquing systems may be characterized. The set of issues addressed in this work are: the user community that the system seeks to assist; the role of the critiquing system in its environment; the extent of the system's advisory capability; the method of critique generation; the set of intervention strategies determining the intrusiveness of the critic; the capability of adapting to needs of the specific user or work situation; the existence of a user goal modeling or user task modeling capability; and the design of an appropriate user interface to communicate the critiques.

The answers a critiquing system provides to these questions defines its method of interaction with the human user. This chapter concludes with a description of SEDAR along these dimensions.

According to Silverman [1992a], the goal of expert critiquing systems is not machine deduction of how to perform tasks, but of machine-assisted human induction/deduction. In other words, such systems recognize that humans tend to err in problem-solving situations and help the human reach improved task performance along some set of metrics. Common metrics include: decreased error rate, decreased task completion time, increased optimization of a set of design quality measures. How an expert critiquing system goes about achieving these goals is less clear. Critiquing systems vary according to their intentionality, critiquing strategy and ability, and a host of other variables. Both Silverman [1992a] and Fischer [1991] provide a set of these variables, which are presented below along with a few additional variables of the author. From reading the literature, it is easy to come away with a sense of the dimensions being boolean in nature, having just one of the two possible values denoted by the characterization. In reality, most systems rarely fall neatly into one category or the other for any dimension; they usually fall somewhere between the two extremes.

## User Community

Determining the experience level of the user community in the task domain is one of the first tasks that a researcher interested in constructing an expert critiquing system must face. The reason for this is that different types of users impose different demands on the critiquing system. To illustrate this, Figure 6 shows a common ontology for the user population based on experience.

Individuals at the "novice" and "intermediate" skill levels are relative newcomers to the task domain and are not yet fundamentally competent in the task domain. Typically, they are people with less than 2 years experience in the domain. They are prone to basic misconceptions, missing knowledge, biases, and mistakes in judgment. Critiquing users at these levels of competence requires in-depth knowledge of the task domain and an ability to diagnose and correct problems in the users' *mental models* [Rook 1993] of the task. Systems that deal with novice and intermediate-level users are also known as intelligent tutoring (ITS) or intelligent computer-assisted instruction (ICAI) systems.

As individuals gain more experience in the task domain, they become "practitioners" and "proficient practitioners." Most of the professionals in a given task domain fall within these two categories. In contrast with novice- and intermediate-level users, practitioners and proficient practitioners are fundamentally competent in the task domain. Their errors are usually due to minor slips, forgetting of knowledge, or biases. Thus the critiquing strategies employed by systems intended for this level of user are very different from those used for novice- and intermediate-level users. We will discuss this issue and its implications in greater depth in the following chapter. If the task is complex (e.g., interdisciplinary in nature, or if knowledge life expectancy is short), unfamiliarity with state-of-the-art practices may also cause errors. Finally, practitioners and proficient practitioners are usually situated in a work environment. As has been seen with the A/E/C community, time and resource constraints may also play a major role in causing errors. SEDAR is intended to support users at this experience level.
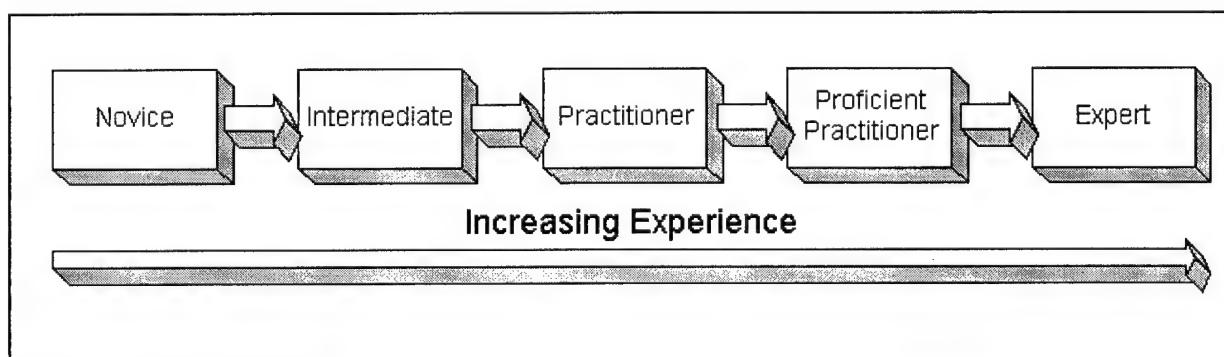


Figure 6. Experience levels.

"Masters" or "experts" are scarce. They are individuals who define and extend the state of the art in their domains. They are capable of considerable introspection about the domain, and are the sources of the knowledge used in expert critiquing systems.

## The Role of Critiquing Systems in Design Environments

The goal of designing a performance-oriented critiquing system is to improve user performance for a task in a work environment. Such systems act as surrogate team members or advisors.

In Figure 7, two computer critics are members of the team designing the product. The critics examine the product design and interact with members of the design team to improve user performance. Possible definitions of "improving user performance" are that the end product has fewer errors, better optimizes measures of design quality, or is completed in less time. Such systems are created with the quality of the final product as the primary goal.

Educational critiquing systems play the role of teacher to their students (Figure 8). Such systems are concerned more with correcting the basic conceptual biases of the
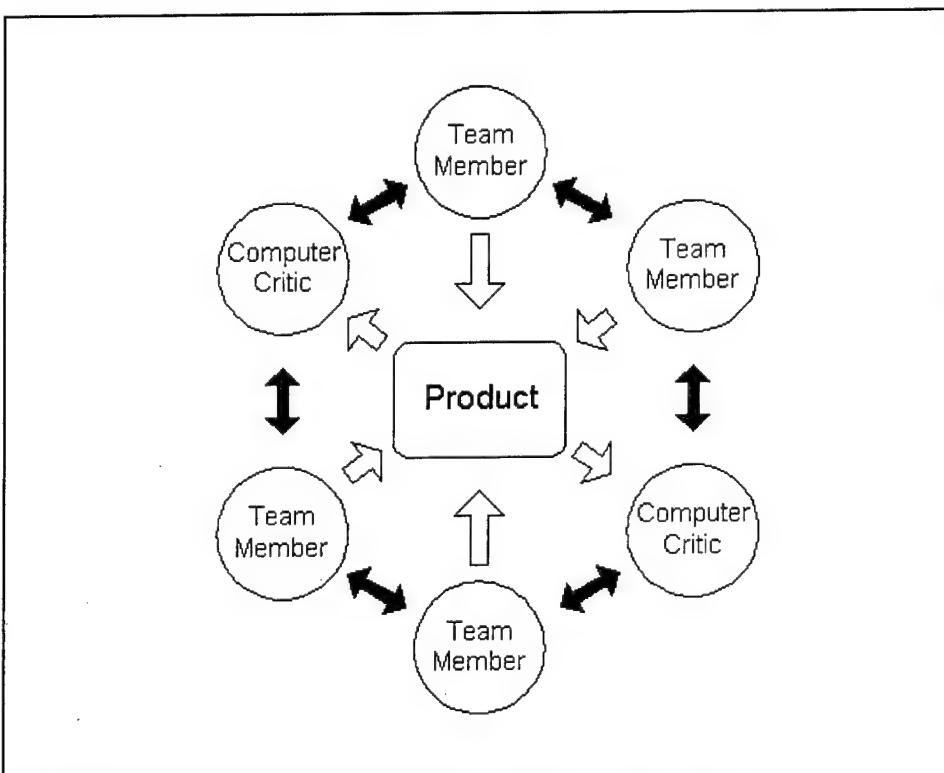


Figure 7. Performance-oriented critiquing system environment.
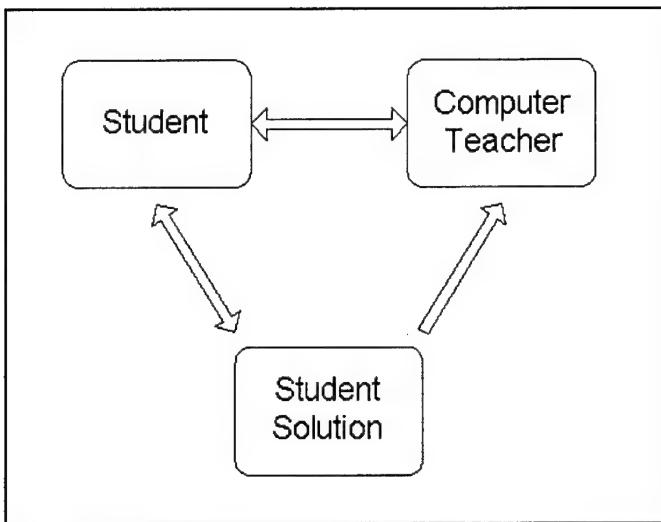
**Figure 8. Education critiquing system environment.**

user and with providing missing knowledge that the user lacks. These types of systems may have more elaborate explanation (tutoring) schemes that allow them to explain the high-level motivations for critiques that they generate. Intelligent computer-assisted instruction (ICAI) systems and intelligent tutoring systems (ITS) fall into this category of critiquing systems.

Most systems usually fall between these two categories. It would be in the best interest of performance-oriented systems to consider the lessons learned from educational systems, and vice versa. Despite the specific examples that were presented, the two roles are certainly not mutually exclusive. Every expert critiquing system must sometimes take on the role of teacher if the critiquing situation demands the role. In a performance-oriented environment, promoting correct problem-solving behavior will lead to improved performance in subsequent tasks. Thus it is worthwhile to explain the motivations behind critiques and advice given by the system. SEDAR is more of a performance-oriented system than an educational system. It is intended to be situated in a work environment where it can provide support to roof designers at the practitioner and proficient practitioner levels of experience.

## Advisory Capability

All critics have the ability to detect errors or suboptimal features of the user's product. After the problems have been detected, critics may rely either on user actions to resolve the problems, or they may suggest alternatives to the user's solution. These types of critics are called solution-generating [Fischer 1991]. Solution-generation is one way of providing constructive criticism to the user. A common term to describe solution generation in a design domain is *design suggestion*.

SEDAR currently supports the error detection advisory capability and a primitive design suggestion capability. It can suggest simple changes (or equivalently, detect object omissions) in the roof design in the form of *shadow objects*. This issue is discussed in more detail in Chapters 6 and 8. Future versions of SEDAR will support more sophisticated solution generation techniques.

## Method of Critique Generation

There are two approaches to generating critiques, differential and analytic. In a differential critiquing system, the system uses a problem-solver to generate its own solution and compares it against the user's solution. The differences are catalogued and shown to the user. Differential critiquing systems have the advantage in that the system knows the optimal (or close-to-optimal) solution to the problem and has found all the differences between its solution and the user's solution. When solutions for a problem are radically different (but equally valid), there may be a problem if the system generates its solution independently of the user. In this case, the solutions may be irreconcilable—the system cannot explain why the user's solution is suboptimal, nor can it explain the differences between the two solutions.

Analytic critics check solutions with regard to predefined features using pattern-matching techniques and expectation-based parsers. An example of this is a system that encodes its "checking knowledge" as a set of condition-action rules culled from handbook and human expertise. One of the problems with analytic critiquing systems is knowing what subset of the rules to apply for each critiquing episode.

The knowledge encoded in SEDAR's knowledge base was taken from low-slope conceptual constructibility model by East, et al. [1995]. This model itself had its basis in the National Roofing Contractors Association's (NRCA) *Roofing and Waterproofing Manual for Low-Sloped Roofing* [1985]. Since the original form of the design codes was that of condition-action rules, SEDAR is currently an analytic critic. As SEDAR's advisory capability grows to include solution generation, its critique generation strategy will also include differential techniques.

## Intervention Strategies

The link between man and machine is especially critical in the expert critiquing paradigm. Careful thought must be given to when the system generates and displays its critiques. One important distinction is between "active" and "passive" critiquing strategies. Passive critics are user activated. Code optimizers, spell checkers, and debuggers are all examples of passive critiquing strategies. Recent research has focused on active critiquing strategies that act whenever events warrant their application. Thus active critiquing strategies are often incremental in nature, which means that they are triggered by user design actions as the design process unfolds. Another view is that these systems try to provide critiques and advice at the appropriate time rather than wait for the user. Intervening immediately after a user action that results in a suboptimal design has the advantage that

the problem situation is still fresh in the user's mind. Furthermore, immediately correcting significant flaws reduces or perhaps eliminates the need for expensive redesign later. The disadvantage of an active critiquing strategy is that too-frequent or improperly timed critiques may disrupt the concentration of the user. Passive critics are user-activated and are often batch-processing critics, which act on a specified set of design objects. Batch-processing critics have the advantage of critiquing "completed" (at least in the mind of the user) portions of the design.

Silverman [1992a] provides an additional distinction among active critiquing strategies. These strategies can be divided according to their timing with respect to tasks in the problem-solving process. For the domain of roofing design, an example of a task is the design of the rainwater drainage subsystem. Before-task criticism is intended to prevent errors before they occur. For example, the system may remind a building designer of an unfinished portion of the building that should be completed before working on other areas. During-task criticism occurs during the performance of the task. In this case, imagine a critiquing system monitoring a user's actions and providing critiques for errors as they occur. Finally, after-task criticism is provided after the task is completed. Taken individually, each of the active critiquing strategies has its flaws. It is often difficult to decide what to show a user to prevent error; displaying too much information confuses the user, while displaying too little information reduces the effectiveness of the preventive strategies. During-task criticism may cause users to respond to only the local criticisms instead of rethinking the entire design task, resulting in more suboptimal designs. A system using only after-task criticism strategies produces critiques at a time when the user has already committed to a solution and may be reluctant to deviate from that solution. The effectiveness of criticism is severely reduced in this type of environment. Systems must combine active critiquing strategies to produce a system that provides criticism at the appropriate time.

SEDAR supports both passive and active intervention strategies. The two active, incremental strategies are the before-task error-prevention critic and the during-task error-correction critic. The design review critic is a passive, batch-processing critic.

## Adaptation Capability

Every user has different preferences and skills. A critiquing system needs the ability to adjust its criticism strategies to the needs of the specific user. Repeatedly displaying a critique that the user has decided to reject is unacceptable, as is

redisplaying critiques that the user already understands. Fischer [1991] calls this ability the system's adaptation capability.

When discussing adaptation capability, it is useful to differentiate between systems that are adaptable and those that are adaptive. In adaptable systems, the user can explicitly change the behavior of the system by setting its parameters or by altering its knowledge content. An example of an adaptable system is one in which the user can add new design objects, relations between the objects, and design rules. Another example is a system with a "reject critique" button that permanently removes a critique.

Adaptive systems automatically change their behavior based on their observations of user responses to their critiquing strategies. Building on the last example presented above, if an adaptive system observes the user always rejecting critiques based on a specific rule, it may remove the rule from future critiquing actions.

SEDAR is currently adaptable with respect to its critiquing components. Users have full control over the activation of its active intervention strategies—they may turn off particular critiques, particular rules in the knowledge base, and even sets of related rules. As discussed in Chapter 10, a critical future improvement for SEDAR is to add the capability of allowing user-created objects, relations, rules, and goals.

## User Modeling Capability

One approach in dealing with the issues described above is to provide the system with a model of the user's goals or of the design task or both (Figure 9). For the
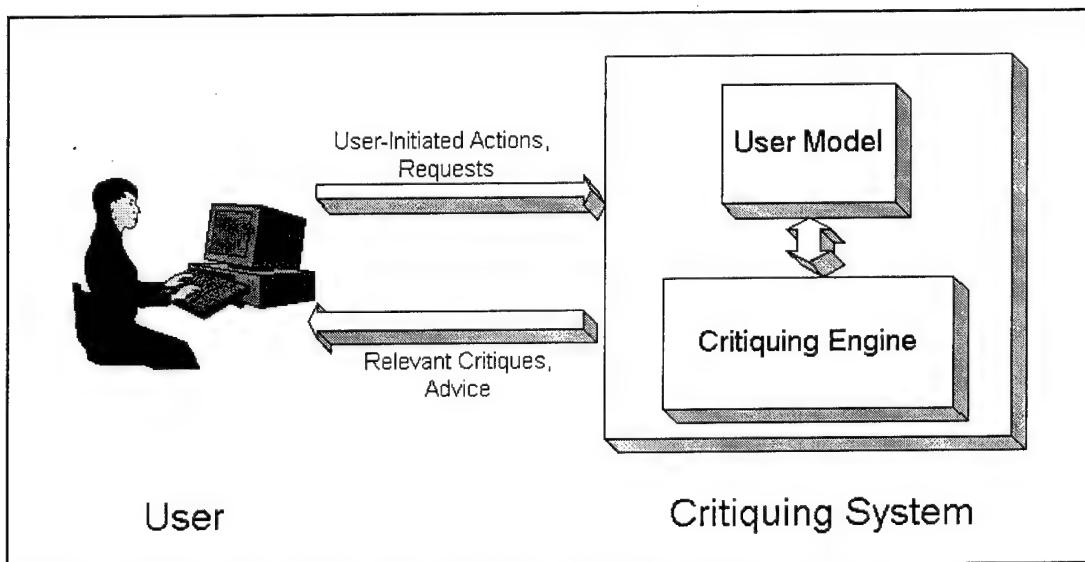


**Figure 9. Adding a user model to the critiquing interaction model.**

purposes of this report, the user's goals are the requirements that the artifact design must satisfy. Examples of user goals for a roof design are watertightness, structural integrity, and aesthetics. Models that represent user goals are called user goal models.

The user's tasks constitute the processes by which the design is created to meet the user's goals. User tasks for the roofing domain would include drainage system layout, equipment flashing design, and roof-mounted equipment layout. Models that represent the user's design tasks are called user task models. For the purposes of this report, these two types of models are more generally called user models, and a system which uses one (or both) of these models is said to have a user modeling capability. Whatever the type of model, the system uses the represented knowledge about the user to adapt its advice to best suit the user's needs.

Silverman calls this "deep" knowledge of the user and the task environment. Recent research in the human-computer interaction domain [Rook 1993] has focused on how systems can foster the development of the appropriate mental model [Rook 1993; Canas 1994] of a problem-solving activity. In particular, Rook and Donnell [1993] concluded that subjects working with expert systems developed personal problem-solving spaces primarily through the construction of a mental model of the expert system task performance. The transfer of information between machine and human is performed via the system's explanations of critiques. Therefore, a user goal modeling or user task modeling capability is important to expert critiquing systems for several reasons. First, a cognitive model allows greater coherency of system actions—by understanding the high-level goals and motivations of experts, the system can provide advice relevant to the user's current goals. Second, such systems have the ability to transfer good cognitive models of the problem-solving process to the user. By providing a complete and correct mental model for problem-solving for a particular domain, the system can generate goal-level and other motivational explanations. Third, modeling provides a framework for critique timing and intrusion, particularly if the modeling encapsulates knowledge of problem-solving tasks. Finally, modeling may reduce the runtime complexity of the system. This is a side-effect of the first point above; a system may be able to constrain its own inferencing relative to the user's focus of attention.

SEDAR's major contribution to the field of expert critiquing systems is in this area. The DTM is a hierarchy of design tasks created from protocol analyses of expert roof designers. Since the roof design task is heterarchically decomposable, we have also modeled the dependencies between the various subtasks. During each incremental critiquing episode, the DTM is used to pick a subset of the design codes in the knowledge base to apply to the existing design. Thus the DTM allows SEDAR to

present the most appropriate critiques relative to the user's focus of attention at any point in the design process.

## The User Interface

The last issue considered is that of the human-computer interface. Although the other dimensions may be well conceived, a critic may still be ineffectual if a proper interface is not developed. Critiquing systems are collaborative, and a great deal of communication must occur across the human/computer boundary. Thus the interface issue is of great concern to the developer of an expert critiquing system. Early expert critiquing systems [Kelly 1984; Miller 1986] used text-based methods to report their results. Often textual displays are too wordy, providing too much information in a manner that is difficult for people to assimilate.

Instead, developers of recent critiquing systems have realized these shortcomings and have worked to develop better models of interaction. Often this means a combination graphical/textual display of the artifact being designed and the critiques generated by the system. Rook and Donnell quantify the benefits of using a graphically-based critiquing paradigm versus a solely textual one [Rook 1993].

SEDAR uses a graphical/textual method of communicating its critiques. It is embedded within a commercial CAD application (AutoCAD[*]), and displays its messages directly on the user's design window. The user may directly manipulate objects on the design (e.g., resize, delete, move). A full explanation of the user interface is in Chapter 8.

## Describing SEDAR Along the Above Dimensions

SEDAR is a performance-oriented critiquing system intended for use with designers at the practitioner and proficient practitioner levels of expertise in roofing design. As such, SEDAR makes several assumptions, of which the two most prominent are mentioned here. The primary assumption that SEDAR makes is that the user is qualified to maintain control of the problem-solving process. From discussions with experienced roof designers, it was found that the order of design might vary according to the individual designer and the particular problem. An example of this is in the interaction between the drainage system design and roof-mounted equipment layout.

---

[*] AutoCAD is a registered trademark of AutoDesk, Inc.

In most cases, the latter task is heavily constrained and is performed first. However, in some cases the drainage system is more heavily constrained and is performed before mechanical equipment layout. Thus SEDAR is flexible with respect to the user's problem-solving preferences. The second assumption is that the user is qualified to judge the correctness and salience of critiques. Thus critiques and the rules that generate them can be overruled by the user. The above issue is related to the adaptation capability of SEDAR. SEDAR is adaptable with respect to the activation of its critiquing components; the user may toggle its critiquing strategies, goals in the DTM, rules in the knowledge base, and even specific critiques. Currently, SEDAR does not have the capability to allow the user to enter new objects, rules, or goals; this is a planned enhancement for the system (Chapter 10).

SEDAR is able to anticipate and to detect errors. It also can generate simple design suggestions. These advisory capabilities are based on the set of rules extracted from roofing handbooks created by organizations such as the NRCA. The condition-action nature of these rules casts SEDAR as an analytic type critic. Future improvements may include a differential analyzer to initiate deeper critiques and design suggestions. The error prevention critic is an active, incremental, before-task critiquing strategy. Its role is to cue users of possible constraint violations by drawing "off-limits" areas directly on the design. The incremental, during-task error correction critic is less intrusive than the error prevention critic but is still active. Its task is to notify the user of existing problems on the design relevant to the last object placed on the design. Instead of directly displaying the constrain areas on screen like the error prevention critic, the error correction critic displays a message regarding the critiques, and displays them on user request. Finally, the design review critic is a passive, batch-processing, after-task strategy. Activated explicitly by the user, the design review critic discovers all design flaws with respect to the design tasks described in the DTM. For example, the user may perform a review on the drainage system design, or on the layout of mechanical equipment. The critiquing strategies are described in greater detail in Chapter 6.

The DTM allows SEDAR to interpret the user's actions on the design and to adjust its critiques according to what it perceives as the user's "focus of attention." A description of the use of user goal models in previous critiquing systems is found in the following chapter; their differences with SEDAR's DTM are described in Chapter 4.

Finally, SEDAR uses a direct iconic manipulation interface embedded in a design environment for roofing. Graphical/textual critiques are displayed directly on the roof design. Comparisons between SEDAR's method of critique display and previous tools in the design/review (construction management) field are described in the following chapter.

# 3 Related Work

The primary focus of this chapter is to describe and to contrast previous expert critiquing systems to SEDAR. Besides characterizing each system according to the issues presented in the previous chapter, the interesting or novel features of the systems are also described. The critic descriptions are roughly in chronological order starting with the oldest system first and working up to recent systems.

Many design systems that fall outside of the expert critiquing paradigm have also influenced this work. In the second part of this chapter, the relationships between the work on SEDAR are compared to other types of systems:

- Intelligent CAD Systems
- Intelligent Interfaces
- Assisted Design Generation Systems.

## Expert Critiquing Systems

The purpose of this section is to provide a survey of the developments in the expert critiquing field over the past decade. Figure 10 is a timeline of some of the influential systems introduced in the 1980s and 1990s. Since the low-slope roofing domain of SEDAR falls within the broader scope of engineering design, the bulk of the discussion below will deal with systems involved in subareas of engineering design.

### CRITTER

CRITTER [Kelly 1984] was one of the earliest expert critiquing systems for engineering design. The system evaluated circuit designs for functional correctness, operating speed, timing, robustness, and sensitivity to changes in device parameters. CRITTER operated in the classical batch-processing style—it received a circuit schematic, a set of behavioral specifications, and a set of characteristic signals it should accommodate. The schematic was then evaluated using various circuit analysis techniques. The results included information about whether the circuit would work and by what margins. The text-based design critique listed the unsatisfied behavioral specifications along with a brief explanation of the problems.

WIZARD 1983 — ONCOCIN

PROLOG Explaining — 1984 STEAMER

CRITTER 1985

ATTENDING — 1986

WANDAH 1987

Design Advisor — 1988 CRITIC

ACTIVIST

LISP-CRITIC

JANUS
FRAMER

CLEER 1989

1990

COPE
TIME 1991

1992

HYDRA

1993

1994

1995 — SEDAR
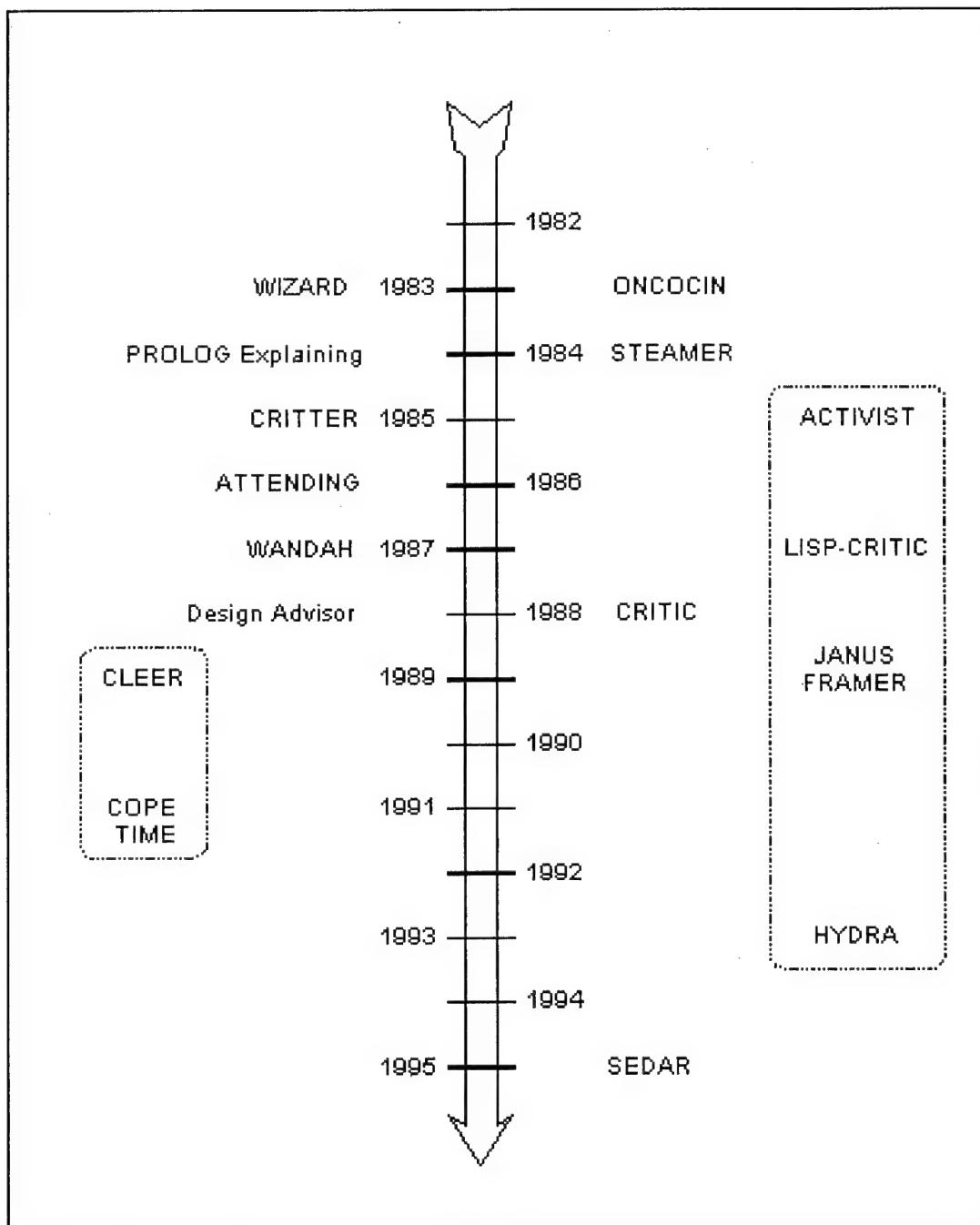
**Figure 10. Timeline of critiquing systems.**

## *CRITIC*

CRITIC [Spickelmier 1988] is an expert critiquing system that looks for errors and "bad design style" in circuit designs. It is intended as both a performance-oriented and educational tool.

CRITIC requires a low-level description of a circuit and a knowledge base that describes a particular design-style and technology. In particular, the knowledge

base contains a set of primitives that describe the circuit elements that CRITIC will find directly in the circuit description. The knowledge base also contains structures, which describe interconnections of primitives and other configurations to be found by CRITIC. Structures are used in CRITIC to provide a framework for the critiquing process. Specifically, CRITIC first maps the circuit design's primitives to defined structures, and then uses the structure and primitive representation to generate its critiques. The use of structures in the critiquing process is similar to a primitive form of plan recognition. Since CRITIC does not require a functional description of the circuit to be given as part of the input, it tries to construct its version of the functional description of the circuit from the design itself. One can view this as a bottom-up approach to describing user intention. The contrasting top-down approach occurs when a system receives a functional description (the user's goals) of the artifact as input. In this case, the system tries to match the observed user actions to its functional description.

CRITIC uses a library of design rules to check for design errors. Critiquing operates in a passive, after-task mode—the user submits a circuit design to CRITIC and receives a list of errors when the run is completed. CRITIC is integrated with a design environment, which allows for a graphical/textual display of its critiques. Before the critiquing run, the user may annotate the circuit design to mark areas of the circuit where certain errors should be ignored.

### ATTENDING

ATTENDING [Miller 1986] critiques plans of anesthetic management for patients. As input, ATTENDING takes a list of a patient's underlying medical problems, the patient's planned surgical procedure, and a user-generated anesthetic plan specifying the various agents and techniques used for general or regional anesthesia. Since the complete anesthetic plan is expected as input, ATTENDING is a batch-processing critiquing system. As has often been the case with the first generation of critiquing systems, ATTENDING is also a passive system—the critic is invoked explicitly by the user.

One of the early systems that performed differential critiquing, ATTENDING analyzes the given plan with respect to minimizing risk to the patient. Besides evaluating the risk of the given anesthetic plan, ATTENDING is able to generate and to evaluate alternative plans. Thus ATTENDING was one of the first solution-generating critiquing systems. Specifically, ATTENDING maintained a hierarchical decomposition of the decisionmaking process for generating anesthetic plans. The approach to generating alternative plans is to consider smallest global changes first,

and then to move on to larger and larger changes if the decrease in risk is not adequate. ATTENDING uses a natural-language interface.

### JANUS

JANUS [Fischer 1991] is an integrated design environment based on the critiquing paradigm. Its purpose is to allow a designer to construct residential kitchen floor plan layouts and to help them learn general principles underlying such constructions. JANUS consists of two subsystems, JANUS-CRACK and JANUS-VIEWPOINTS.

JANUS-CRACK is a knowledge-based design environment supporting the construction of kitchens using domain-specific building blocks called design units. In JANUS-CRACK, the user is presented with a graphical/textual interface that allows direct manipulation of the design. JANUS-CRACK applies design principles from its knowledge base to the existing kitchen layout. There are "hard" principles like building codes and safety standards, and "soft" principles like user preferences. The design principles are captured as condition-action rules; thus JANUS-CRACK can be considered an analytical critiquing system. Its critics are active in that they are triggered by user actions on the design. For example, if a user moves a stove on the kitchen layout, the stove critic activates and tests the new stove location for compliance with stove-related design principles. The critics in JANUS-CRACK are during- and after-task critics because they are activated in reaction to a specific user action. Finally, the critiques generated by JANUS-CRACK are textual, but are linked to issue-specific hypertext information in JANUS-VIEWPOINTS.

JANUS-VIEWPOINTS is a hypertext-based system containing recorded examples of good general principles of kitchen design. When the user clicks on a textual critique generated by JANUS-CRACK, JANUS-VIEWPOINTS is activated and displays the argumentation associated with that critique. Thus JANUS is characterized as having limited solution generating abilities. While JANUS does offer constructive advice on where design units should be placed, this advice is not problem-specific, only domain-specific.

JANUS is an adaptable system that allows the user to modify the design environment by adding, deleting, and altering design units, critic rules, and relationships. Additionally, users can add their own examples of good design to the JANUS-VIEWPOINTS hypertext system.

## CLEER

Zhou et al.'s CLEER system [1989] is designed to help improve electromagnetic compatibility among shipboard topside equipment and their associated systems. In particular, the purpose of CLEER is to determine the feasibility of a location where an antenna will be placed. The objective that CLEER tries to achieve is to avoid unintended blockage or distortion of the radiation patterns, to realize maximum intended range, and to avoid being a source of electromagnetic interference (EMI) to other electronic devices.

CLEER draws its domain-specific knowledge from a set of databases and knowledge bases. Two databases store the ship description and the characteristics of equipment installed on ships. Constraints, heuristics, and analogical information is stored in three separate knowledge bases. Constraints are "hard" rules of thumb that must be followed, or infeasible equipment configurations will result. Heuristics are "soft" constraints in that they do not have to be satisfied to make an antenna arrangement feasible. CLEER uses these constraints in the search for the optimum location for an antenna. Finally, the analogical knowledge base stores past cases of antenna placement.

CLEER first displays a graphical layout of the topside of the ship. This information includes the architectural structures as well as the equipment mounted on the ship. The user is presented with several options to change the equipment arrangement of the ship. One such action is to add a new antenna to the topside layout. When an antenna type is selected, CLEER searches its analogical knowledge base, based on the type of antenna and warship for similar antenna settings. If there is no past experience, the user chooses a location on the ship. CLEER then accesses its constraint knowledge base to see if the arrangement is within a desirable limit with respect to safety requirements, warfighting capability and overall performance of the system. The antenna location is considered acceptable if all constraints associated with the antenna are satisfied. If the location is rejected, the reasons to justify the conclusion are displayed on the screen. CLEER then tries to find a better location for the antenna. Its heuristic search algorithm (hill-climbing) is directed by the "soft" constraints in the heuristics knowledge base. If a better location is found, it is displayed as part of the explanation, along with the critical heuristics that CLEER used.

## LISP-CRITIC

LISP-CRITIC [Mastaglio 1990] is designed to support LISP programmers. Users ask LISP-CRITIC for suggestions on how to improve their code, and the system

replies with transformations that make the code more cognitively efficient (easier to read and maintain), or more machine efficient (code runs faster or uses less memory).

LISP-CRITIC is thus a passive, batch-processing system. The critic's condition-action rules contain code patterns as their antecedents and code transformations as their consequents. Besides the rule library, LISP-CRITIC has a LISP domain model, which represents LISP in terms of its underlying concepts and basic functions.

Of particular interest for this report is that LISP-CRITIC has a user-modeling component. The model is used to customize explanations, determine which subsets of rules to fire for each individual, and to provide the information to place users in a tutoring environment. The user modeling component of LISP-CRITIC serves as an interesting contrast to the DTM of SEDAR, and thus will be explained in greater detail.

**LISP CRITIC's User Modeling Component.** The user modeling component consists of access and update methods and a database containing individual information about users. The database contains the following information about the individual user:

- The *User Model* is a representation of what the user knows about LISP concepts and functions.
- The *Dialog History* is a list of the explanation episodes that the user has seen already.
- The *Preference Record* is a list of favored LISP functions and "turned-off" LISP-CRITIC rules.
- The *Code Analysis* contains two parts: (1) a profile of rules that have been fired in the past and their acceptance or rejection by the user and (2) statistical data.

Thus the user model represents the critic rules, functions, and concepts that the system thinks the user knows. The model is built up over time (critiquing episodes) by the update methods.

**Use of the User Modeling Component.** The user modeling component is used to tailor explanations to the particular user. For example, when a critique is generated and the user requests an explanation of the rule, the LISP domain model is checked to find the dependencies between the critiqued concept and subconcepts (more basic concepts). The explanation module queries the user model for subconcepts with

which the user is unfamiliar. Additional explanation is then provided for those subconcepts.

Unfortunately, Mastaglio does not explain how the user modeling component is used to determine which subsets of rules to fire for each individual or how the knowledge may be used in a tutoring environment. A logical conclusion one may reach from Mastaglio's discussion is that the set of rules "turned off" by the user are not used in future critiquing episodes.

## COPE

COPE [Silverman 1992a] is a critic programming environment. The programmer uses a graphically-based editor that helps author bias/error identification and pre-ventative/corrective/repair strategy knowledge bases. COPE's function library holds high-level building blocks for implementing different critiquing functions such as showing hints, analogs, or defaults to prevent errors; and error checking, explanation generation, and repair functions to detect and resolve errors. The programmer defines a decision network that describes when to activate the programmed critics.

COPE is the first expert critiquing system shell. It improves upon existing expert system shells by providing specialized support for critiquing systems. It provides the framework for programming systems to support users with various levels of expertise, from the novice to the proficient practitioner. While COPE uses a mixed media interface, it is not clear whether the resulting critiquing systems are mainly textual or graphical in nature.

## HYDRA

HYDRA [Fischer 1993] is the successor to JANUS. It consists of four components: a construction component that serves the principal medium for the design, a specification component that allows designers to specify design requirements, an argumentative hypermedia component, and a catalog component that contains a collection of previously constructed designs. HYDRA also has three distinct critiquing strategies. Generic critics apply domain knowledge concerning desirable spatial relationships between design units. Specific critics detect inconsistencies between the design and its specifications. Finally, interpretive critics are topical groupings of critics and design knowledge. The user may define an interpretive critic to critique the design from a particular perspective. The example that Fischer provides is that of a resale-value perspective. This critic would include domain knowledge pertinent to homeowners concerned about their home's resale appeal.

Like JANUS, HYDRA's critics are during- and after-task active critics. Critiques are textual in nature and shown in a message window below the construction component. Like LISP-CRITIC, HYDRA has a user-modeling component. The user model is the set of functional requirements specified by the designer, so the critiques produced by the specific critics are tied to the representation of the articulated goals of the design project.

## Other Types of Related Systems

### Concurrent Engineering Systems

**ICADS**. The ICADS testbed [Pohl 1992] focuses on the development of a cooperative computer-aided design environment. ICADS uses a cooperative decisionmaking model with a blackboard control system and several independently executing domain experts and Intelligent Design Tools (IDTs). An IDT is a design solution evaluator. Based on their respective input templates, IDTs respond directly to changes in the current state of the design solution initiated by the designer in the computer-aided design (CAD) drawing environment. With minor exceptions, their evaluative capabilities are limited to quantitative analysis of the physical parameters of the building design.

The ICADS model also includes a conflict detection/analysis module that makes countersuggestions (based on the evaluation results to the suggestions from the designers) and a message router that receives and sends information to all participants in the dialog.

ICADS thus fits within the framework of concurrent engineering systems, which seek to fuse different perspectives on a product into a unified environment. Agents, both human and computer-based, represent the different viewpoints on design. The equivalent of the agent in the ICADS system would be the IDT.

Concurrent engineering issues were considered in the development of SEDAR. For example, the SEDAR architecture allows multiple agents to lend critiques from different perspectives on the roof design. Unfortunately only a single agent dealing with constructibility assessment is currently implemented. Also, SEDAR has no principled way of dealing with conflict detection and resolution. Another view of SEDAR is as a single agent for a concurrent engineering platform. SEDAR would lend critiques on the flat and low-slope roof design of a building design being developed in a concurrent engineering platform.

*ACE*. ACE [Case 1994] is a concurrent engineering platform being developed at the U.S. Army Construction Engineering Research Laboratories (USACERL). Like ICADS, ACE brings together multiple perspectives on a design in a principled fashion through the sharing of data structures. ACE also provides a means for conflict detection and conflict resolution through regulated information sharing. When a change is made on the design, only the set of agents relevant to the change are notified for approval. A version of SEDAR is under development as an agent in the ACE environment.

### Intelligent Interfaces

CHECS (Chemical Engineering CAD System) [Goodman 1990] is a plan-based intelligent interface for computer-aided chemical process design. Of interest is its use of plan recognition to support its interface tasks. Specifically, CHECS observes a sequence of the designer's actions to discover the underlying plan, drawn from a plan library, of a chemical plant design. Once the plan is selected, the next logical step in the plant design process is inferred from the chosen plan.

SEDAR may be described as a plan-based intelligent interface. It accepts user actions as input, forms beliefs of the user's intentions and goals, and provides assistance based on those beliefs. However, as is discussed in Chapter 6, SEDAR's use of the DTM is both more powerful and more flexible than the plan recognition system of CHECS.

### Assisted Design Generation Systems

VEXED [Steinberg 1992] is a design-assistance system developed to model the design process as progressive top-down refinement and constraint propagation. Refinement of a design involves structural decomposition by breaking a module, a group of components being viewed as a functional block, into its subcomponents.

VEXED takes as input a circuit design problem represented as a "black box" module with specifications on various features of its inputs and outputs, such as their datatypes, values, timing, and encoding. Modules may be broken into smaller, semi-independent submodules through the use of a set of refinement rules (expressed as condition-action rules). Primitive modules are associated with known components of the target technology. The output of VEXED is a fully completed design of the artifact. However, VEXED is not an automated design generation system. It requires input from the user to decide which uncompleted module to refine in each successive step.

VEXED operates in a refinement cycle. At the start of each cycle, VEXED displays a menu of the modules remaining to be refined, and the user selects one to refine. VEXED then finds all the rules that apply to the module, and displays them in a menu. The user selects one of the rules, and VEXED applies the rule to the existing design. Thus the user is responsible for strategic decisions, while VEXED takes care of the detailed manipulation and constraint propagation needed to carry out the user's decisions.

If no rules apply to a module, or if user wants to do something not enumerated in the menu, he or she can use a graphical editor to decompose the module manually into submodules and their interconnections. LEAP (Learning Apprentice) generalizes the manual decomposition step into a new rule that can be used in subsequent refinement cycles. VEXED records the refinement steps as an annotated tree-like design plan that shows the final design plan for the artifact.

## Chapter Summary

The first section of this chapter described a set of expert critiquing systems organized in a roughly chronological order. Each system made its own contributions to the expert critiquing field and had its own particular strengths and weaknesses. Early critiquing systems used passive, after-task, batch-processing, textual critiquing strategies. Recent critiquing systems have developed active, during-task, incremental strategies embedded in graphical environments. Another trend in the growth of expert critiquing systems is the realization that multiple types of critiquing strategies are needed for different critiquing situations. Recent systems are able to generate critiques based on different perspectives.

One weakness seen throughout the history of development of expert critiquing systems is how to use a user goal model or a user task model to guide the critiquing process and how to make the model an integral part of the control of the critiquing strategy. Thus SEDAR's use of the DTM may be viewed as an effort to fuse the user model with system control. The benefits of this fusion are that the system will be able to offer better advice at more appropriate times to the user. Throughout the rest of this report, the systems discussed in this chapter will be used to compare and contrast with SEDAR's design philosophy and implementation.

# 4   Research Contributions

The primary contribution of this work is the use of a task model of design (or a user
task model as described in Chapter 2) to flexibly control the critiquing process. The
presence of a user model allows the system to adapt to the needs of the particular
user. Without user models, critical issues such as the appropriateness of the content
of critiques throughout the problem-solving process, the timing of incremental
critiquing strategies, and the ability to critique the user at higher levels of abstrac-
tion are difficult to address.

The use of the DTM in SEDAR differs from previous efforts implemented in expert
critiquing systems in two ways. First, SEDAR uses an explicit task model of design
(the DTM) that expresses the dependencies between the tasks that a user must
accomplish during the design process. This model is very different in structure than
the user models employed by previous critiquing systems like LISP-CRITIC and
HYDRA. This issue is discussed in greater detail below and in Chapter 6. Second,
the use of the DTM allows SEDAR finer-grained control over what critiques are
generated at each point in the problem-solving process. This ability is essential in
incremental critiquing systems like SEDAR, where the system must be able to adapt
the content of its advice as the design changes over time. This issue is discussed in
greater detail in Chapters 6 and 8.

Najem [1993] wrote in his doctoral dissertation:

> Critiquing is a very challenging problem if the problem-solving method is non-
> deterministic. Critiquing then requires that the system be able to comprehend
> the problem-solving actions of a community of experts in the field...The
> challenge is how to find a single representation that can simulate the coherent
> plan of a single expert when solving a problem, yet at the same time encompass
> all reasonable plans when acting as a critic.

One way to address the challenges that Najem describes is the use of an explicit user
goal model or user task model as a control structure in the critiquing engine. The
key point is that the user model must be flexible in terms of representing the
problem-solving behavior of experts.

Most expert critiquing systems do not have a user-modeling component. (A few that do were discussed in Chapter 3.) LISP-CRITIC and HYDRA both have user-modeling components. LISP-CRITIC's modeling component consists of the set of concepts and functions that the system believes the user understands, a set of preferred functions, a set of "turned-off" rules, and a profile of past user actions. During problem-solving activity, the "turned-off" rules are removed from LISP-CRITIC's working rule set. The system's beliefs about which concepts and functions are known to the user are used to derive explanations for the critiques. HYDRA's user model is the set of kitchen requirements that the user defines throughout the design process. Thus HYDRA employs a user goal model as defined in Chapter 3. Of HYDRA's critiquing strategies, the specific critics are used to assess whether the design meets the user-established requirements.

The DTM is a representation of an experienced roof designer's task structure. The activation pattern of the DTM determines which subset of rules in the knowledge base are applied to the existing design. Thus finer-grained control is attained over the content of the critiques offered to the user in comparison with the techniques used by LISP-CRITIC and HYDRA.

The activation pattern of the DTM is updated continuously during the problem-solving session to reflect the system's beliefs about the user's changing design tasks. Due to the richness of the knowledge of relationships between tasks, SEDAR is able not only to apply knowledge from tasks in the designer's immediate focus of attention, but also from those tasks that should be considered. When critiques are to be generated, the state of the DTM is used to define the appropriate subset of the knowledge base to apply to the existing design. At the same time, the DTM is used in a flexible way. The user is not compelled to alter their problem-solving activity to fit the system model, which is a possible problem with CHECS and is discussed in Chapter 6.

SEDAR is used as a testbed for various critiquing strategies. Currently SEDAR has three critiquing strategies (error prevention, error correction, and design review) that are very different from each other in terms of timing, intrusiveness, and intent. In particular, the error prevention strategy, an active, before-task critic, was developed and tested. Few error prevention critics have been implemented because of the many difficult questions raised by such critics. First, what type of assistance should be provided? JANUS provides examples of good kitchen design that may help to prevent future errors from occurring. SEDAR takes a simple but more principled approach to generating preventative assistance. As the user adds to the design, SEDAR illustrates the "off-limits" constraint areas on the existing design. These

constraint areas are generated from the set of design codes in SEDAR's knowledge base.

An evaluation of SEDAR was conducted to assess the effectiveness of each of these strategies in a performance-oriented critiquing system interacting with roof designers of the intermediate, practitioner, and proficient practitioner levels of expertise. Their comments about the strategies are discussed in Chapter 10.

Another contribution that SEDAR makes is its method of displaying constraints between design objects. SEDAR uses a combination graphical and textual display method. The graphical portion of the critiques consist of constraint areas displayed directly on the design window.

# 5  The Roof Design Domain

This chapter will introduce fundamental definitions and concepts relevant to the design of flat and low-slope roofs. A basic understanding of the roof design domain will in turn facilitate explanations of the motivations, operations, and problem-solving structure of SEDAR. A thorough discussion of the roof design domain is beyond the scope of this work. Readers interested in learning more about roofing design are referred to the work by Griffin [1982].

The first section describes the requirements and goals that a roof design must satisfy. The two most important requirements are that the roof maintain both watertightness integrity and structural integrity throughout its lifespan. Beyond these first two issues are a number of interrelated requirements that the roof designer must address. The major components of a flat or low-slope roof are presented in the second section. The process of creating a roof that meets the requirements specified in the first section begins with a high-quality roof design, which is argued in the third section. The fourth section describes the roof design task and how the natural structure of problem solving in roof design influences the system structure.

## Roof Requirements and Goals

As stated above, two important requirements for a roof are that it maintains watertightness integrity and structural integrity throughout the roof's expected lifespan. Loss of watertightness integrity, resulting in water leakage, may be due to a number of causes. Improperly protected wall edges and roof penetrations may allow water to enter the roof membrane. Besides being an added load capable of sagging a roof, the ponding of water as a result of insufficient drainage capacity or an incorrectly sloped roof serves to expose all weaknesses in the roof's waterproofing defenses. Ponding is especially dangerous for the class of flat and low-slope roofs described in this paper. To help prevent conditions like ponding, significant portions of roofing handbooks are devoted to establishing guidelines for proper rainwater transmission and collection in the roof field. For example, the NRCA roofing handbook

establishes a minimum deflection of 1/2 in. per foot* for low-slope roofs. The task of designing the water transmission and collection subsystem will be discussed later in this chapter. The roof must also maintain structural integrity in the face of both normal and adverse environmental conditions. Year-round, the roof must support the weight of not only the equipment mounted on the roof but also rainwater, snow, and even wet leaves. Strong winds may also cause an uplifting force on the roof. Since various materials expand and contract at different rates due to temperature, the roof design must be able to accommodate at least a small amount of movement of the building. However, the roof is still responsible for a great deal of the dimensional stability of the building itself and should not be overly flexible.

A number of factors besides the above requirements influence the roof designer. Some of the factors that the roof designer must consider are shown in Figure 11.
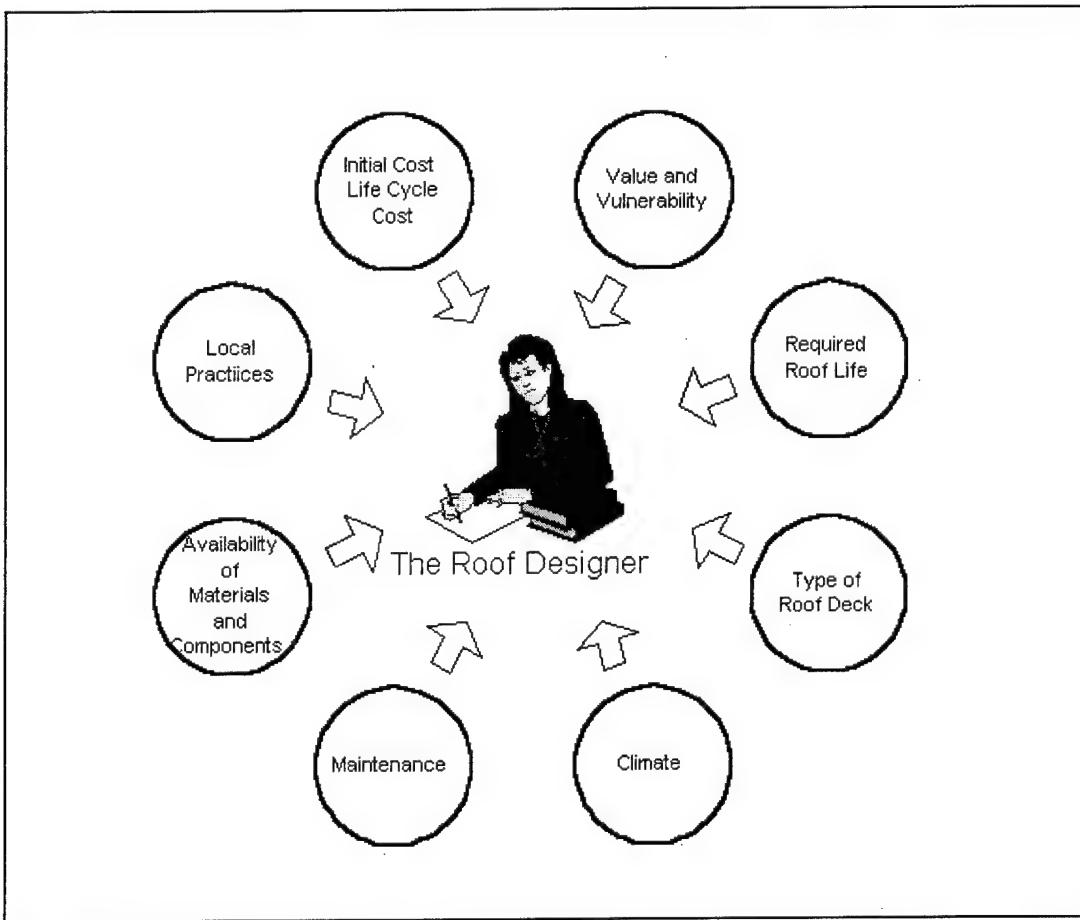


**Figure 11. Other requirements for roofs.**

---

* 1 in. = 25.4 mm; 1 ft = 0.305 m.

## Major Roof Types, Subsystems, and Components

A roof system is an assembly of various functional subsystems composed of interacting roof components designed, as part of the building envelope, to protect the building interior, its contents, and its human occupants from the outside environment. This section describes the major types of roofs, functional subsystems, and constituent roof components that are found in a typical flat or low-slope roof system. Within SEDAR, design is performed at the roof component level. Designers may select components (or design objects) from a predefined palette and place them on the roof design.

### *Major Roof Types*

This research considers two of the most common types of roofing systems, built-up and single-ply. Of the two, built-up roofing has historically been by far the most popular roof; however, in recent years single-ply roofing has made significant advances in the roofing market. Figure 12 shows that built-up and single-ply roofs have three major layers: the structural deck, the thermal insulation, and the membrane. The primary difference between built-up and single-ply roofs is the membrane material and construction, described below:

- The Structural Deck—The structural deck transmits gravity, earthquake, and wind forces to the roof framing. The deck often provides the slope for runoff transmission and also serves as an anchoring surface for roof components.
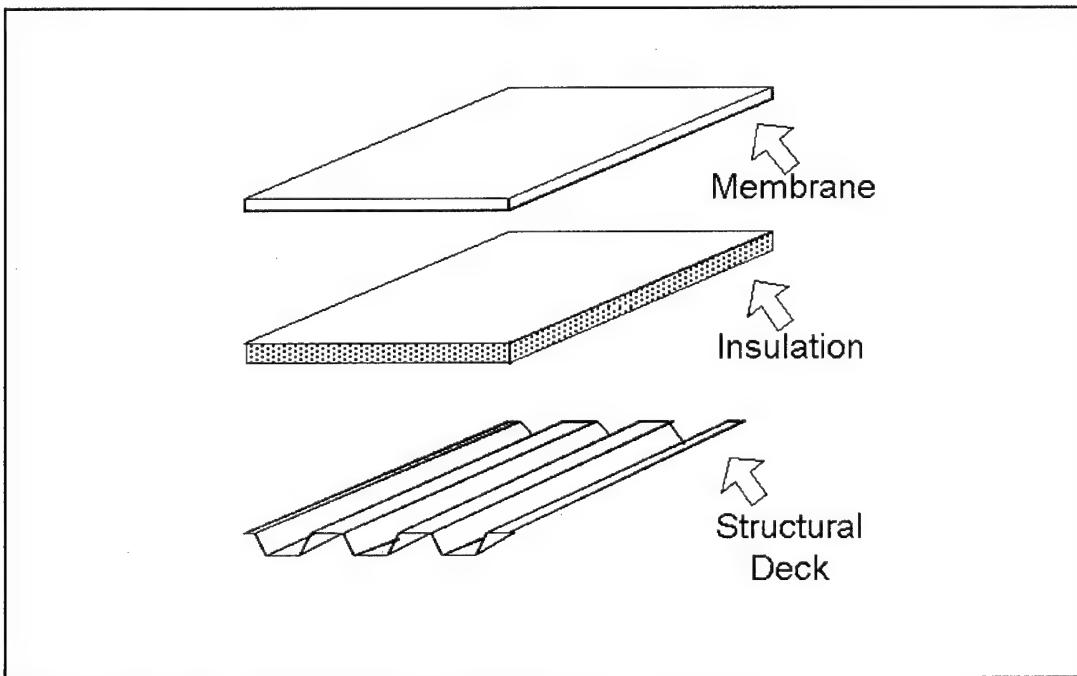


**Figure 12. The three layers of a flat or low-slope roof.**

Finally, the deck maintains the dimensional stability of the entire roof system, and is fire-resistant. Decks are made from a variety of materials, including wood (both timber and plywood), ribbed steel, and various forms of concrete.

- The Thermal Insulation—The thermal insulation's primary purpose is to decrease heating and cooling costs. Additionally, it provides horizontal shearing resistance and acts as a substrate upon which the roof membrane is applied.

- The Membrane—Built-up roofing membranes consist of alternating layers of felts and bitumen, with a final surfacing layer on top. The membrane forms a semiflexible covering that completely covers the roof surface and shields the roof components underneath it. Bitumen (usually coal-tar pitch or asphalt) serves as the waterproofing agent; the felts serve to stabilize and strengthen the bitumen layers. The surfacing layer is normally gravel, crushed rock, or blast furnace slag. Its purpose is both to ballast the membrane against uplifting forces and to protect the bitumen from life-shortening solar radiation.

In single-ply roofing, the membrane is usually a single layer of rubber-like material instead of the felt/bitumen sandwich construction used in built-up roofing. One such type of material is ethylene propylene diene monomer (EPDM). Despite having the same basic construction as described for built-up roofs, single-ply EPDM roofs have many different requirements than built-up roofs. For example, since there is no surface aggregate to ballast the roof, an adhesive must be used to secure the membrane to the insulation layer. Additionally, since the EPDM material is intrinsically vapor retardant as well as water retardant, a vapor retarder is not needed in the roof design.

### Some Major Roof Subsystems

The roof system itself may be divided into various functional subsystems, which are assemblies of interacting roof components. The study of the different subsystems is important because human designers tend to perform roof design in terms of designing individual functional subsystems. Examples of functional subsystems involving the roof system are the roof frame, the rainfall transmission and collection system, the heating, ventilating, and air conditioning (HVAC) system, and the roof ventilation system. Each of these functional subsystems involves a subset of the total set of roof components. Often the design of these subsystems is heavily constrained by the existing building design. For example, the roof frame system, which provides the structural support for the roof, is not created by the roof designer; its design is the task of the structural engineer. Thus the columns, beams, joists, and other components of the roof frame are already established by the time the roof design is addressed. The same constraints hold true for the HVAC system—large roof-mounted components such as air-handling units, exhaust fans, and power vents are usually tied to specific locations.

One roof subsystem that is less constrained and is a major responsibility of the roof designer is the rainfall transmission and collection system, also known as the drainage system. The designer must decide between having an interior or a peripheral drainage system. In an interior drainage system, the drainage of the roof is achieved by a system of slopes that lead rainfall into the interior of the roof, where roof drains attached to leaders (pipes) conduct the water down through the building interior. Leaders for interior drainage systems are almost always located at columns. Peripheral systems direct water away from the center of the roof to its edges, where water is conducted away from the roof surface by gutters. Both types of drainage systems have advantages and disadvantages.

Interior drainage systems are often found on roofs with elevated edges (parapets) that serve to shield the roof-mounted equipment from visual sight. Interior drain pipes are heated by the building interior and continue to conduct water through cold winter weather, while peripheral drainage systems are subject to ice damming and metal distortion from repeated freeze-thaw cycles. The roof drains of interior drainage systems may become blocked with debris (e.g., leaves fallen from trees) and may cease to provide adequate drainage.

Water transmission within the roof field, whether using an interior or a peripheral drainage system, is achieved in flat and low-slope roofs by gently sloping the roof surface. For flat roofs, this sloping is achieved by tapering the thickness of the insulation boards underneath the roof membrane. For low-slope roofs, variations in the heights of vertical structural members (i.e., columns) is the primary source of slope. The inverted pyramidal pattern with the low point at the center containing an interior roof drain, shown in Figure 13, is an often-used sloping pattern. Another common sloping pattern is the saddle, shown in Figure 14, used in decks sloped only in one cross-section. Roof-mounted equipment also form barriers (or dams) to roof surface water flow. Crickets are required on the high side of such equipment to divert water flow around these obstacles. Figure 15 shows a properly placed cricket.

### Major Roof Components

In this section the various roof components of functional subsystems are described. Figure 16 shows a cut-away view of a flat or low-slope roof with an interior drainage system. In addition to providing more detail for the major roof layers (structural deck, insulation, and membrane), it also shows roof-mounted equipment typically found on flat and low-slope roofs. Definitions of the following roof components are taken from the *Roof Consultants Institute's Glossary of Terms* [RCI 1994].

Source: Griffin 1982. Used with permission of The McGraw-Hill Companies.
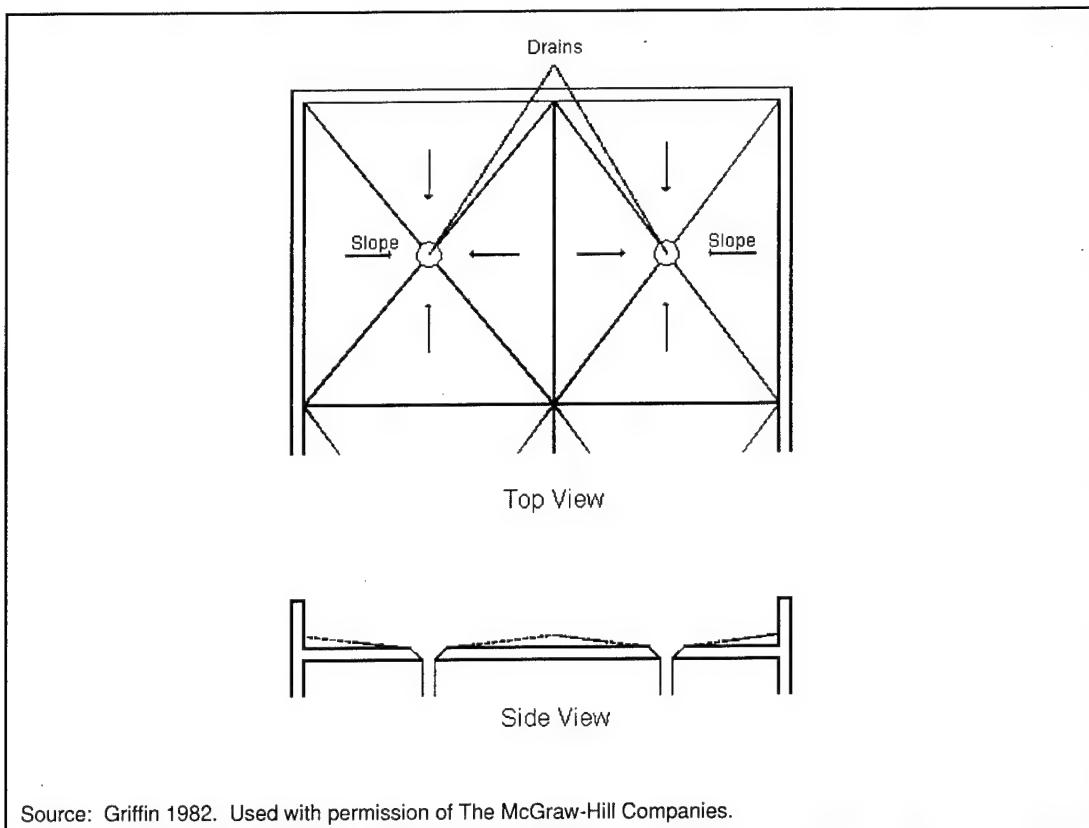
**Figure 13. The inverted pyramid drainage system.**



Source: Griffin 1982. Used with permission of The McGraw-Hill Companies.
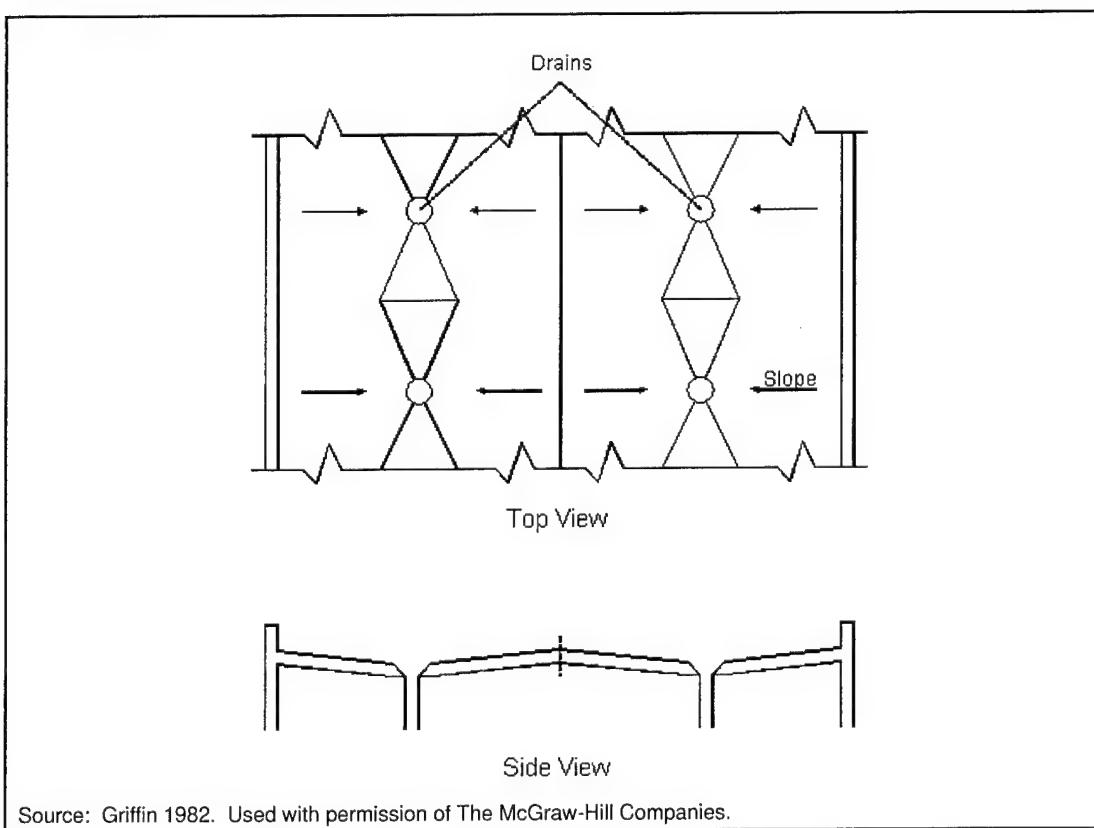
**Figure 14. The saddle drainage system.**
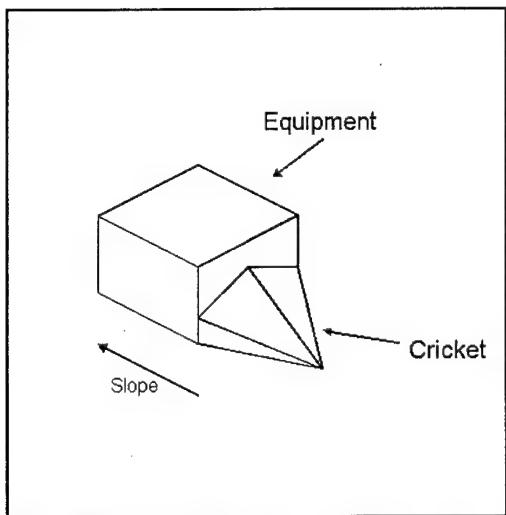
**Figure 15. A cricket on the upstream side of roof-mounted equipment.**

The roof frame's main elements are columns, beams, and joists. Columns are the structural members used in a vertical position to transfer loads from the main roof beams, joists, etc. to the building foundation. Beams are the primary horizontal members subjected to bending loads from the roof. Joists are secondary horizontal members usually laid perpendicular to the beams.

The fluted metal deck lies on top of the roof frame. The rigid insulation lies on top of the deck. Insulation usually comes in rectangular boards, which may be mechanically fastened to the deck. The fastener pattern for the insulation on this roof is shown in the upper left corner of the diagram. This roof has a built-up membrane, which requires ballast for anchorage. This is shown in the lower portion of the roof.

Besides the major components, additional roof components are shown in Figure 16. Discussion of these additional components begins with the sump and continues clockwise around the building.

- Sumps are rectangular depressions in the surface of the roof around the opening to a drain and serve to promote drainage.
- Roof drains are devices that allow the flow of water from a roof area. Often a filter or a grill covers the drain to prevent debris from entering.
- Roof-mounted equipment like mechanical units or air-handling units (not displayed in the figure) provide various functions for the interior of the building. This equipment is often mounted on wood curbs and supported by the underlying roof frame. Figure 16 also shows an example of drain piping or leaders. The drain piping conducts water from the roof drain through the interior of the building. Here, the piping runs down the building wall instead of a column.
- Scuppers are used as overflow relief for interior drainage systems. They are openings through the parapet wall above the roof deck. Aside from the scupper itself, a system consisting of a conductor head, drain leader, and splash block complete the overflow drainage system.
- Moving to the lower portion of the roof, roof hatches allow access to the roof. They are typically hinged panel units that are fastenable and weathertight.
- Pavers or walkways are used to provide paths from the roof hatch to roof-mounted equipment. The roof membrane is easily damaged by foot and repair

ROOFING COMPONENTS
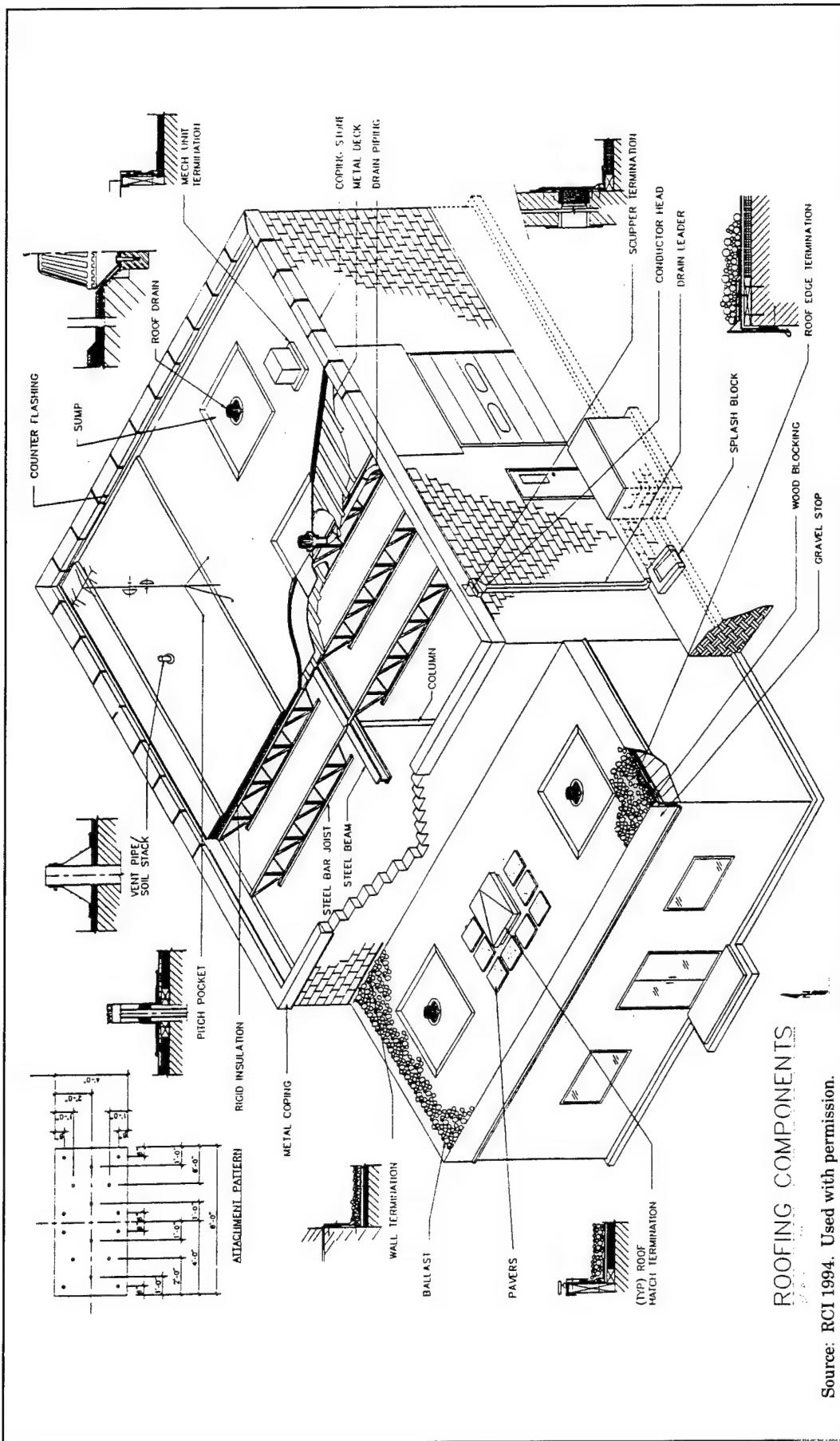
Source: RCI 1994. Used with permission.

**Figure 16.  A cut-away view of a building's roof system.**

equipment traffic, and thus is protected from normal traffic patterns by these components.

• The parapet wall is often covered by a metal coping, which protects the wall from exposure to weather.

Surrounding the building in Figure 16 are numerous detailed views of cross-sections of the roof. These *construction details* specify the roof design at a lower abstraction level. These drawings are typically standardized for the type of roof and the roof component. Sources of construction details include the NRCA roofing handbook and component manufacturers.

## Why Roofs Fail and What Is Done To Prevent Failures

Roof fail for a variety of reasons. A common causality chain [Griffin 1982] may look like the following:

1.  The designer produces a design of marginal quality and skimps on the roofing system specifications and details.

2.  The general contractor, disregarding the roofer's qualifications and ignoring the application technique, selects the low roofing bid and relies on the roofing manufacturer's inspection to verify that the roof subcontractor used proper construction materials and techniques.

3.  The roof subcontractor, if given leeway on the roof design, may opt to select a cheaper way to satisfy the roof specifications.

4.  The manufacturer's inspector, who is often the sales representative who sold the materials to the roof subcontractor, is charged with inspecting the work of the roof subcontractor on whose continued good will the representative depends on for future sales.

Thus assigning responsibility for a roof failure is a formidable challenge. Often several parties are at least partly at fault. However, the roof designer has a responsibility to do all that he or she can to ensure a durable, high-quality roof design. This means creating a good (instead of marginal) complete set of roof drawings and specifications. An example of a high-quality roof design is one in which water shedding is preferred over water resistance as a means of keeping water out of the building. The reason for this is that practicable field techniques of applying a roof often fall far short of the thoroughness and precision found in the laboratory.

Problems ranging from thin spots in the top coat of bitumen to a simple puncture from a dropped tool may result in the failure of the roof membrane. However, if the roof is designed to shed water effectively, it will be able to survive some imperfections without leaking.

The design/review protocol of A/E/C firms is an attempt to improve the quality of roof designs by subjecting the designs to several review sessions when various sources of expertise are used to discover and resolve problems with the roof design. The sources of expertise include paper checklists of common design flaws, handbooks of design codes, and the experience of the reviewer.

## The Roof Design Task and Decomposition

To understand how to improve on the existing design/review process, protocol analyses were conducted of roof designers creating roof plans starting from a blank drawing board. As was mentioned previously, the act of designing a roof is divided into a set of tasks oriented toward creating particular subsystems of the roof system. Subsystems are created by the selection and placement of certain types of roof components (design objects). Different groups of tasks are performed at each stage of the design process (conceptual, intermediate, detailed, and final design). This section provides a description of the decomposition of the roof design task into its major subtasks and the types of interdependencies between subtasks. The contents of the DTM reflect those tasks addressed in the intermediate stage of roof design. A portion of the DTM is shown and described for this stage.

### The Conceptual Stage of Roof Design

During the conceptual stage of roof design, the designer establishes the fundamental characteristics of the roof. Because the roof plan is usually one of the last systems of the building to be completed, the other building systems usually impose constraints on the roof. One of these constraints is the shape or footprint of the roof. Other characteristics are the type of roof (built-up or single-ply), the insulation material, and the type of drainage system (interior or peripheral). The underlying design of the building may dictate what types of equipment will be mounted on the roof and where that equipment will be placed in the roof field. For example, the type and location of air-handling units are often determined from conferences among the design team. The structural engineer then designs the framing plan to accommodate the equipment load at the decided location, and the mechanical engineer designs ductwork to lead to and from the air-handling unit. Thus the location of the air-handling unit on the roof is completely determined by the original collaboration.

Another case of this is the roof deck material, which is often determined by the structural engineer.

### The Intermediate Stage of Roof Design

***Architectural layout.*** After the determination of the basic characteristics of the roof, the designer begins the architectural layout of the roof. During this stage, the designer works with a top-down, two-dimensional view of the roof. The focus of the designer is to lay out the various roof subsystems at the level of major roof components (design objects). Designers focus on two roof subsystems during this stage—layout of roof-mounted equipment and design of the drainage system, although numerous other subsystems also must be addressed during this stage.

In general, the layout of roof-mounted equipment is usually addressed before all other roof subsystems, especially if the existing building design imposes constraints on the roof plan. As discussed in the previous section, the location of air-handling units may already have been decided upon at this stage. The location of other major roof penetrations, such as chimneys, fume hoods, or roof access hatches, may also be similarly constrained. Even less constrained roof-mounted equipment like antennae, roof vents, and moisture relief vents are also addressed early during the intermediate stage of design.

An important task related to equipment layout is that of walkway layout. Each major piece of roof-mounted equipment must be accessible for repair from a roof access hatch or a wall-mounted ladder. Because roof membranes are easily damaged by traffic, the roof designer must lay out a network of plywood, metal, or rubber walkways that interconnect the serviceable equipment and the roof access mechanism. The walkways protect the membrane from abrasions and penetrations resulting from maintenance activities.

The second major subtask is designing the rainfall transmission and collection subsystem. Whether an interior or peripheral drainage system is used has already been determined at this stage. The drainage subsystem involves a number of roof components:

- Layout of sloped areas for proper water flow to drains or to the periphery of the roof, which involves the placement of inverted pyramid, saddle, and cricket type slopes about the existing roof design
- Placement of drains at low points in the roof field
- Design of the overflow relief system, which may include both scuppers at the roof periphery and overflow drains near roof drains away from the roof's edges.

During the intermediate design stage, the roof designer lays out various building subsystems at the same level of detail. Thus there is the potential for many inter-task conflicts (e.g., a walkway strip may interfere with the water flow from a high point on the roof to a low point with a drain). When conflicts occur, the relevant subsystems are usually "tweaked" to resolve the conflict rather than completely redesigned. The ordering of the satisfaction of subtasks is especially important here. By starting with the most highly constrained subtask and then using the result to constrain subsequent subtasks, the designer usually avoids egregious mistakes that necessitate substantial redesign. As has been observed for many types of problem-solving tasks, the human designer first searches for a satisficing solution instead of an optimizing solution.

***How the DTM represents the roof design task.*** The representation of the above roof design tasks and their interrelations is the DTM. Figure 17 shows a portion of the DTM representing a subset of the intermediate stage design tasks (mostly those described above) and the relations between them. The tree is a part-of hierarchy (left to right) of roof design tasks. The root of the tree is the roof-layout task, which is broken into its constituent subtasks, each representing the design of a different roof subsystem. These are in turn divided into more specific subtasks. The leaves of the hierarchy describe the layout of the specific types of roof components which, taken together, form a particular functional subsystem.

The relationships described in the previous part of this section are also represented in the DTM. The fact that the layout of roof-mounted equipment is generally accomplished before addressing other roof subsystems is expressed by the solid arrow between the equipment-layout and the roof-component-layout tasks in Figure 17. Such links are known as before-task relations, and express temporal orderings between the accomplishment of tasks. A similar situation is true for walkway layout. Design of the walkway system cannot begin until the various roof-mounted equipment are placed in the roof field. Thus a set of *before-task* relations exist between the walkway-layout task and the air-handler-layout, chimney-layout, and hatch-layout tasks.

Potential interferences between roof subsystems are marked by the dotted double-arrowed lines (*interferes-with* relations) between tasks in Figure 17. For this example, only the *interferes-with* links involving the drain-layout task are shown. Thus the potential interference between drain layout and walkway layout described previously is represented by the dotted double-arrowed line between the drain-layout task and the walkway-layout task in Figure 17. The drain-layout task also has *interferes-with* relations with the other subtasks of the general equipment-layout task.
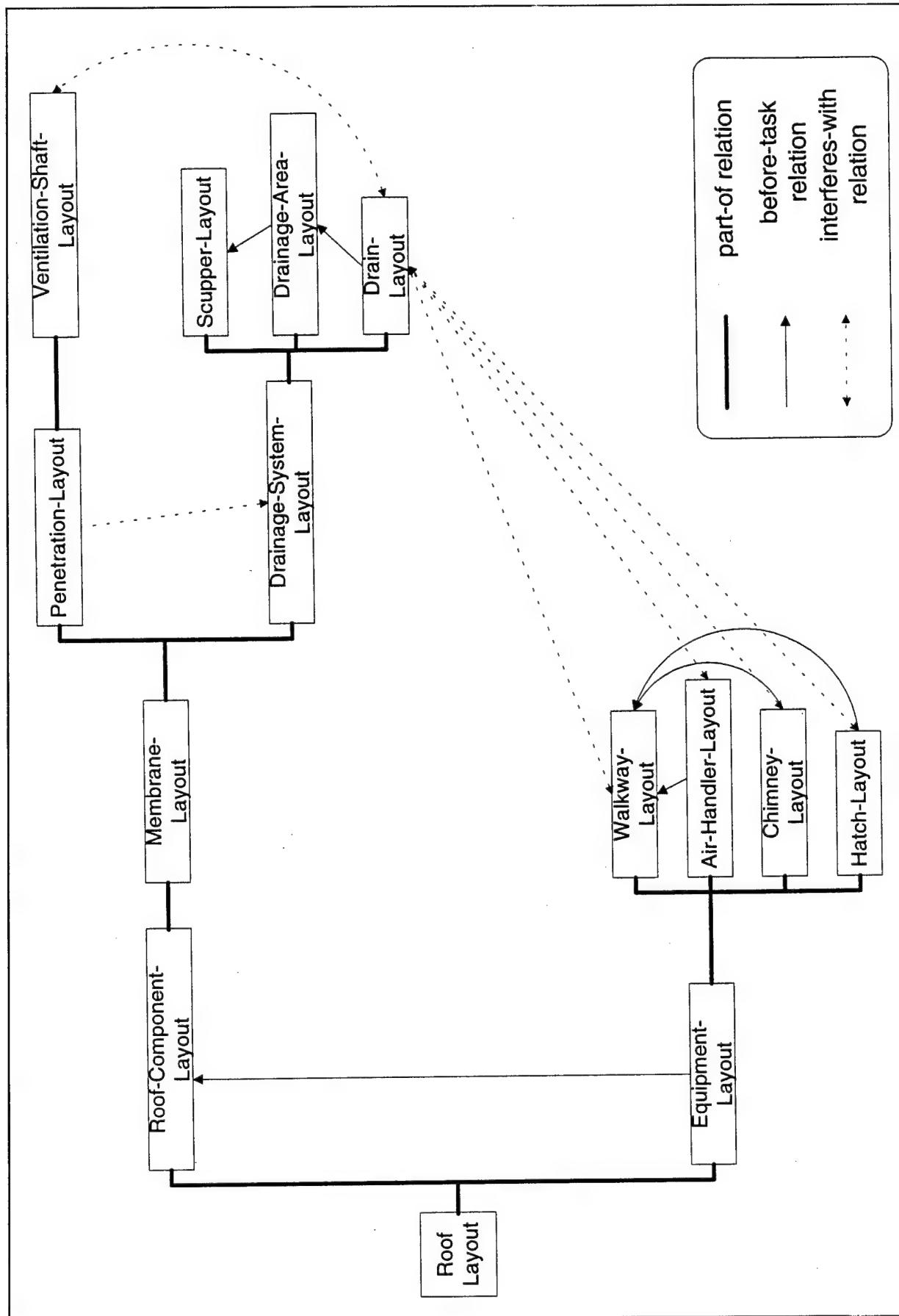
Figure 17. A portion of the Designer's Task Model.

The discussion of the DTM's relation to the task structure of experienced roof designers in this section has focused on the representation of the different types of dependencies between roof design tasks. In the next chapter, the structure and use of the DTM in SEDAR are described in greater detail.

### The Detailed Stage of Roof Design

After the general layout of the roof subsystems is completed, the designer adds additional detail to the drawing during the detailed design stage. For example, the designer will add *construction details* at the side of the drawing showing cross-sections of the roof construction like the subdiagrams of Figure 16. These include roof penetration details for the roof-mounted equipment and flashing details for the roof perimeter. Instead of creating new details for every design, generally accepted details are often copied onto the design from handbooks [Ramsey 1994; NRCA 1985; RCI 1994]. The designer is also responsible for creating a specification document that itemizes the roof components in the design and includes additional comments for the construction team.

### The Final Stage of Roof Design

The final stage of roof design is devoted to holding a final design review and making small changes to the design to accommodate the reviewer's comments. By the end of this stage, the roof design (both the drawings and the specifications) is complete.

## Chapter Summary

Design of the roof system of a building is a complex, multidisciplinary activity. Communication and coordination between the architect and the other members of the design team (e.g., the structural, process, mechanical, and electrical engineers, and the other architects) is necessary to ensure that all the requirements of the roof are met with the roof design. Besides discussing roof requirements, this chapter introduced the major components of the roof and their role in the overall roof system. Finally, observations of human roof designers have provided insights as to how they divide the roof design task and the ordering and dependencies among the resulting subtasks. These insights helped to determine the structure and function of the DTM used by SEDAR.

# 6 System Architecture and Operation

This chapter describes the structure and function of the DTM in the context of the overall system architecture and its operation. Additionally, this chapter establishes a working vocabulary of system components and operations that will be used extensively throughout the rest of this report.

The description of the DTM structure begun in the previous chapter is completed in the first part of this chapter, which deals with the system architecture. The system consists of four major components: the critic management agent, which controls system operation; the blackboard, which contains the system's shared data structures; the critic agents, of which only one, the flat/low-slope roof agent, is currently implemented; and the user interface, which is the medium of interaction between the human user and the critiquing system.

The use of the DTM in the basic operating cycle to direct critiquing in a focused but flexible way has thus far been hinted at but not discussed in detail. The second part of this chapter describes how the DTM is used in the context of the basic operating cycle, called the iterative critiquing cycle. The iterative critiquing cycle is based on the critiquing interaction model presented in Chapter 1.

The third part of this chapter describes the error prevention, error correction, and design/review critiquing strategies in greater detail than in Chapter 2. A full example of the critiquing strategies discussed in this section may be found in Chapter 8.

## System Architecture

The system architecture is presented first to establish a working vocabulary before describing the other aspects of the system in more detail. Conceptually SEDAR consists of four major components: the Critic Management Agent (CMA), the Blackboard, the Critic Agents, and the User Interface. Figure 18 is a diagram of the components and their interrelationships. Each of these components are described in more detail in the following subsections.
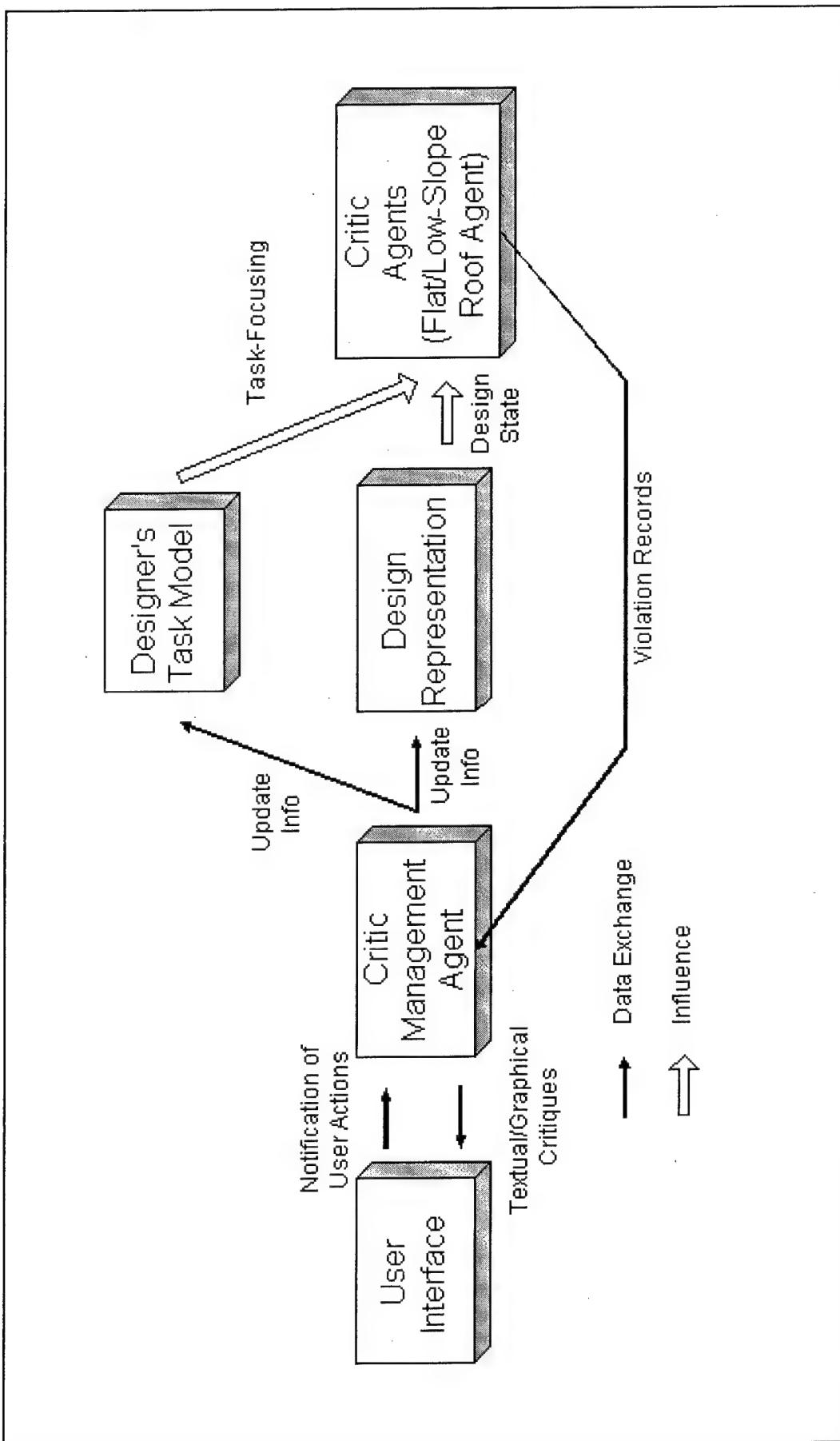
**Figure 18.  SEDAR architecture.**

### The Critic Management Agent

The CMA is the control unit of the expert critiquing system. It receives and interprets descriptions of user actions from the user interface, updates the representations on the blackboard, selects which critiquing strategies to apply, and activates the proper critic agents. Currently the CMA selects from one of three critiquing strategies: error prevention, error correction, and design review. After the critiquing process is finished, the CMA gathers the generated critiques, translates them into critique display descriptions that the user interface understands, and sends them to the user interface. The CMA operates in a loop called the iterative critiquing cycle, which is described in the second part of this chapter.

### The Blackboard

The blackboard contains the shared data structures of SEDAR: the DTM, the Design Representation, the Requirements Hierarchy, and the Materials Hierarchy. While the basic structure of the DTM has been briefly described in Chapter 5, a more complete description of its structure and use are given below.

**The Designer's Task Model.** The DTM is a user task model for roof design based on analysis of experienced roof designers. It is used by the CMA to generate the most appropriate set of critiques for the designer's focus of attention and the existing state of the design. At the same time, the use of the DTM is flexible in that it allows the user to control the problem-solving activity and does not force the user along predefined solution paths. This capability is accomplished by mapping user actions to activation patterns that are the system's beliefs about the current focus of attention of the user on specific roof design tasks. The activation patterns determine the subset of the knowledge base that is applied in the current critiquing episode. Furthermore, the ordering of critiques presented to the user is also influenced by the activation pattern of the DTM, which is updated every time the user performs an action on the design.

The tasks that a roof designer performs to complete a roof design are arranged hierarchically in a tree-like structure, shown in Figure 19. The figure displays the tree from left to right, with the most general task (roof-layout) at the left. This task represents the overall task of roof design. The heavy left-to-right lines connecting the tasks are *part-of* relationships (decompositions) between a task and its subtasks; for example, the equipment-layout task is composed of four subtasks, walkway-layout, air-handler-layout, chimney-layout, and *hatch-layout*. Intermediate nodes represent the task of designing functional subsystems of the roof; for example the *membrane-layout* task encompasses the process of designing the roof membrane
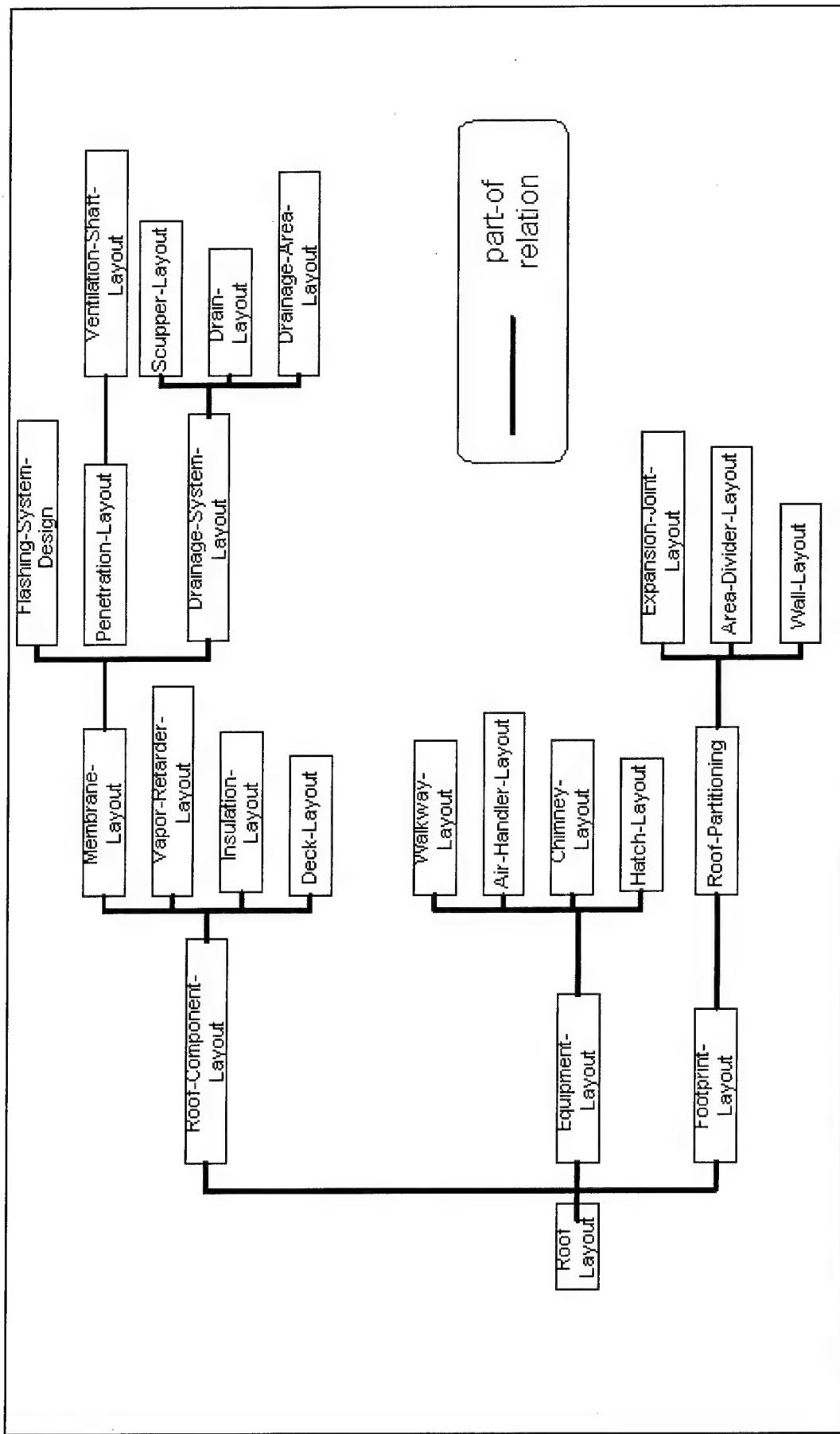
**Figure 19. The Designer's Task Model, viewed along part-of relationships.**

system, which in turn may be decomposed into more specific subtasks. Finally, the leaf nodes of the tree (e.g., scupper-layout, air-handler layout) represent the layout of particular classes of design objects.

Besides the *part-of* relationships, the DTM supports two other types of relationships, represented by the *interferes-with* and *before-task* relations. *Before-task* relations encode observations of expert designers' task orderings, and are usually relations among tasks at the same level in the DTM. These relations are shown with the *part-of* relations shown in Figure 20. The task connected at the tail of the arrow is constrained to be accomplished before the task connected to the head of the arrow. One such relationship is among the footprint-layout, equipment-layout, and roof-component-layout tasks. Expert roof designers usually lay out the footprint of the roof before placing mechanical equipment in the roof field. As has been discussed in Chapter 5, the expert roof designer also tends to lay down large, heavily constrained pieces of equipment before dealing with the drainage system design, flashing design, etc. These types of temporal orderings of tasks are captured by the *before-task* relations.

Knowledge about potential interactions between tasks is represented by *interferes-with* links. Figure 21 shows the *interferes-with* relations for a single task, air-handler-layout. These links are static in nature and encode *a priori* knowledge about which tasks are likely to produce interactions on roofing designs. These relationships exist typically among tasks that are addressed during the same stage of design. For example, during the intermediate design stage the designer will work on many of the immediate subtasks of the equipment-layout task, like walkway-layout, air-handler-layout, etc. Because each one of these tasks may interfere with each other, it is necessary to denote their interdependencies using *interferes-with* links.

Currently only the *interferes-with* and *part-of* relations are used in SEDAR. At this point, the *before-task* relations represent a capability of critiquing the task-ordering performed by the user. This capability has not been implemented because of the desire to make SEDAR as flexible as possible with respect to the individual users' preferences. A principled approach to critiquing user task ordering involves not only the *before-task* relations, but a principled analysis of which functional subsystems of the roof are the most heavily constrained in a given initial roof design specification.

The DTM can have different activation patterns during the design process that encapsulates the system's beliefs about the designer's goals and current focus of attention. Each task may have one of three possible activations. Tasks with focus
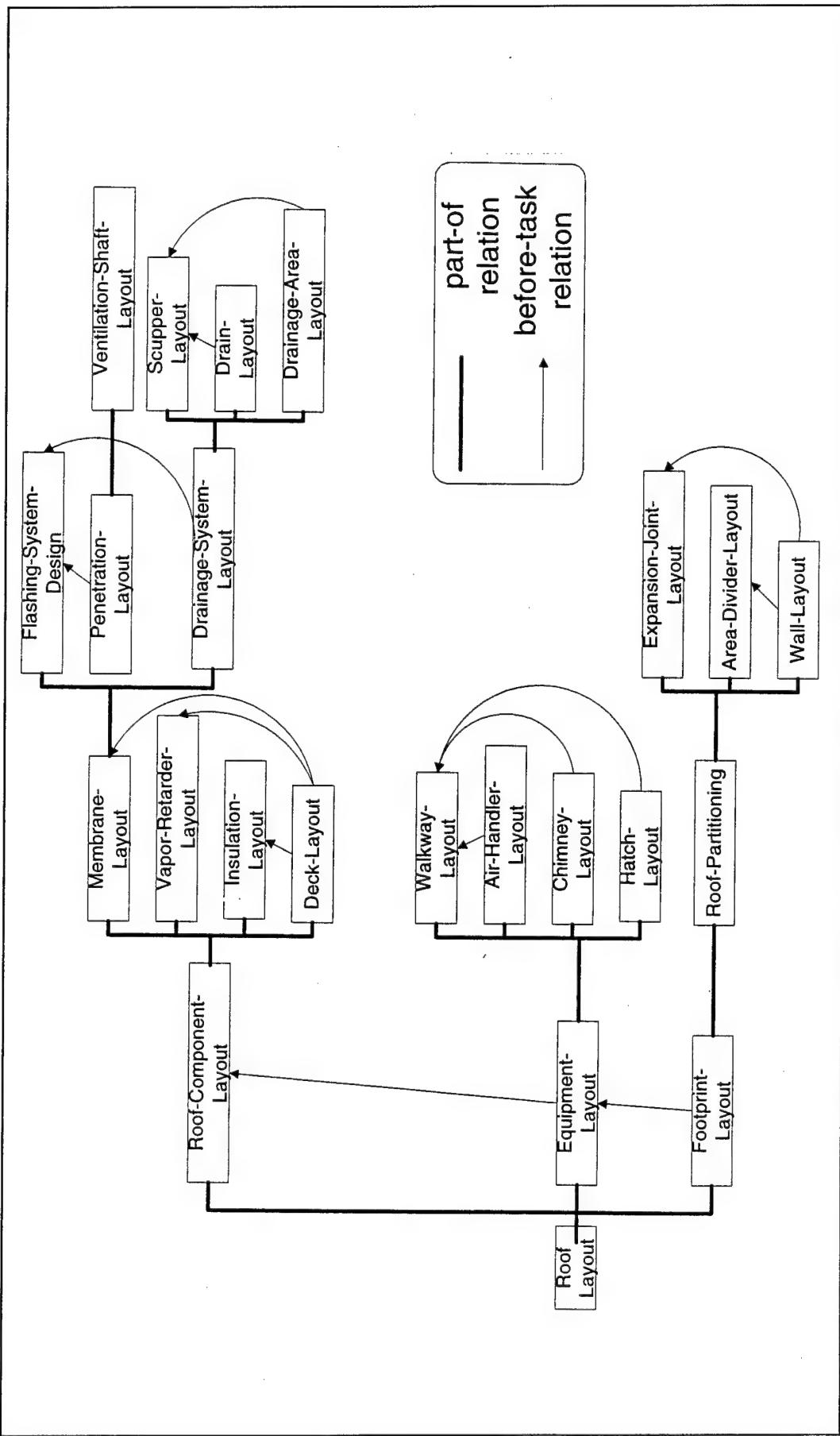
Figure 20.  The Designer's Task Model, viewed along task-subtask and before-task relationships.
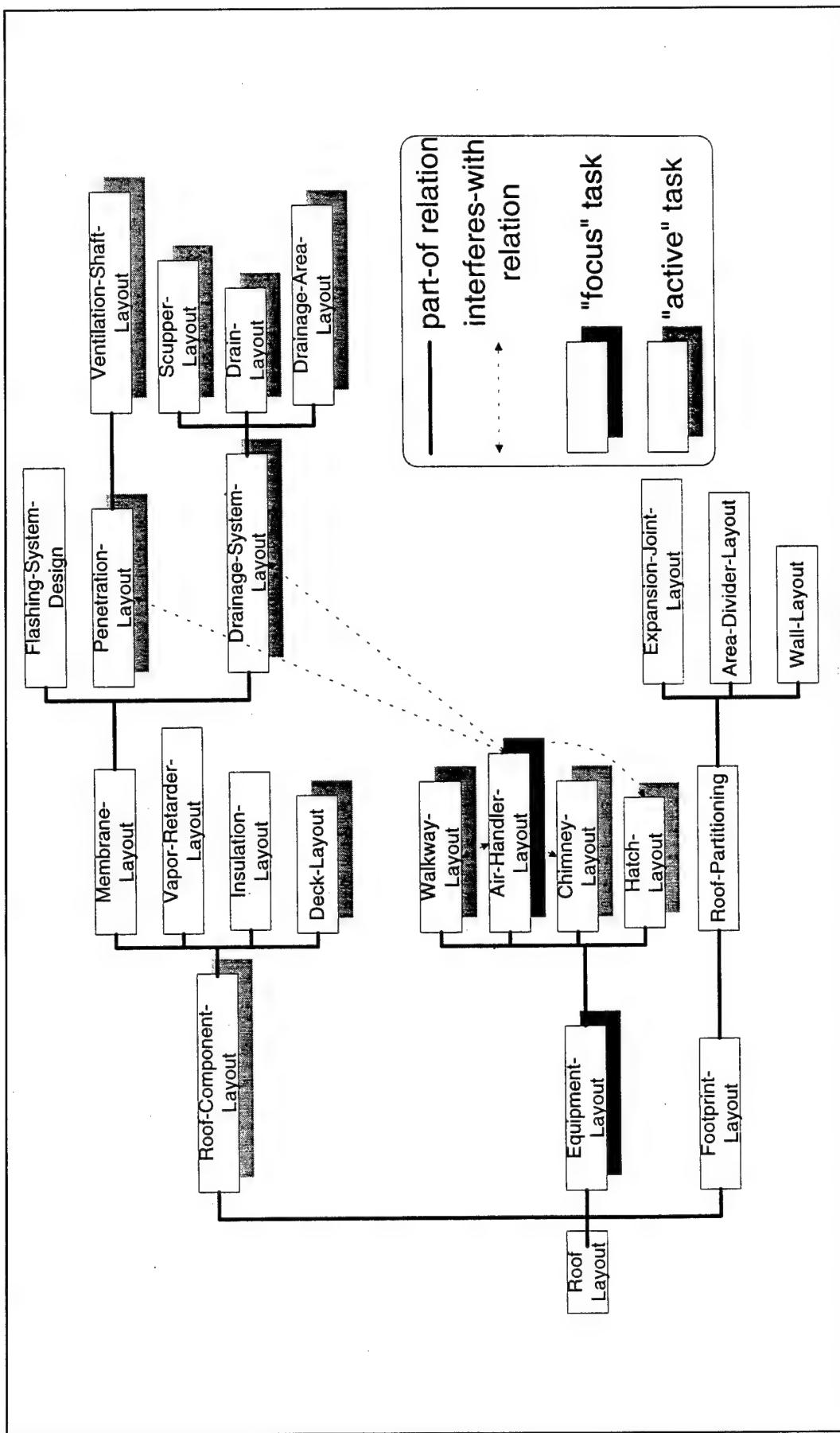
Figure 21. Some task activations and interferes-with relations in the Designer's Task Model.

activation (or focus tasks) are those related to the designer's immediate actions. Each task is linked to a set of design objects, so when the designer selects an object for placement on the roof design, all tasks that had a focus activation previously are changed to being active. The new set of focus tasks is composed of all tasks that include the new object's type in their object set. Tasks that are made active either had a focus activation before or are related to the new set of focus tasks by *interferes-with* relations. Thus the active set of tasks includes those tasks previously within the user's focus of attention and those that should be considered given the current focus. Finally, inactive tasks are those that have not yet been addressed by the designer and are not related to the focus set of tasks by *interferes-with* relations. Figure 21 shows an activation pattern resulting from the selection of an air-handling unit by the designer. The air-handler-layout and equipment-layout tasks are focus, and most of the active tasks are related to the air-handler-layout task by *interferes-with* relations. The deck-layout task is active because it had a focus activation in the past. Thus SEDAR does not use the DTM to direct the user along a solution path. Instead, it interprets and follows the user's problem-solving actions flexibly using the DTM.

The user also has direct control over the state of the DTM. Through the user interface the user may switch tasks on or off. All tasks are initially on, and remain on until the user turns them off. Tasks that are off are not used to generate critiques no matter what their activation (focus, active, or inactive). This ability allows the user to have the final determination of what tasks are involved in the critiquing process. For example, if the user feels that critiquing on a particular task is inappropriate for the existing design situation; he/she is then able to adapt the system to his/her needs by turning the task off.

Each task in the DTM is associated with a set of design codes that pertain to the design of the particular roof subsystem associated with the task. After the activation pattern for the current user action has been established, all of the design codes associated with the focus and active tasks form the subset of the knowledge base applied to the existing design. Since the subset of rules follows the changing activation patterns of the DTM, the critiques generated by this process are closely related to the user's "focus of attention."

After critiques are generated by applying the subset of design codes to the existing design, the critiques are ordered according to the DTM's activation pattern. Critiques related to focus tasks are placed before those related to active tasks. This ordering ensures that the critiques most salient to the user's current focus are shown first.

An interesting comparison may be made between SEDAR's use of the DTM and the plan-recognition system CHECS. In CHECS, the user's actions were used to find the most appropriate plan from a plan library. If the user's actions could not be explained by any plan in the library, CHECS would not be able to continue with its collaboration process. SEDAR's flexible use of the complete DTM may be interpreted as being equivalent to having a maximal set of plans given the component design tasks. Figure 22 shows this comparison. The individual plans of the CHECS plan library may be thought of as paths through the DTM. Thus Plan 1 in the plan library corresponds to the highlighted "Plan 1" path through the DTM.

Under this interpretation, the DTM of SEDAR also has the ability to represent multiple interacting individual plans in CHECS (Figure 23). The individual plans **A-B-C** and **A-D-G-J** are represented as a single task activation pattern in the DTM.

***The Design Representation.*** The Design Representation is SEDAR's internal representation of the design. It consists of a set of design object instances and the semantic links between the design objects. The design representation captures the
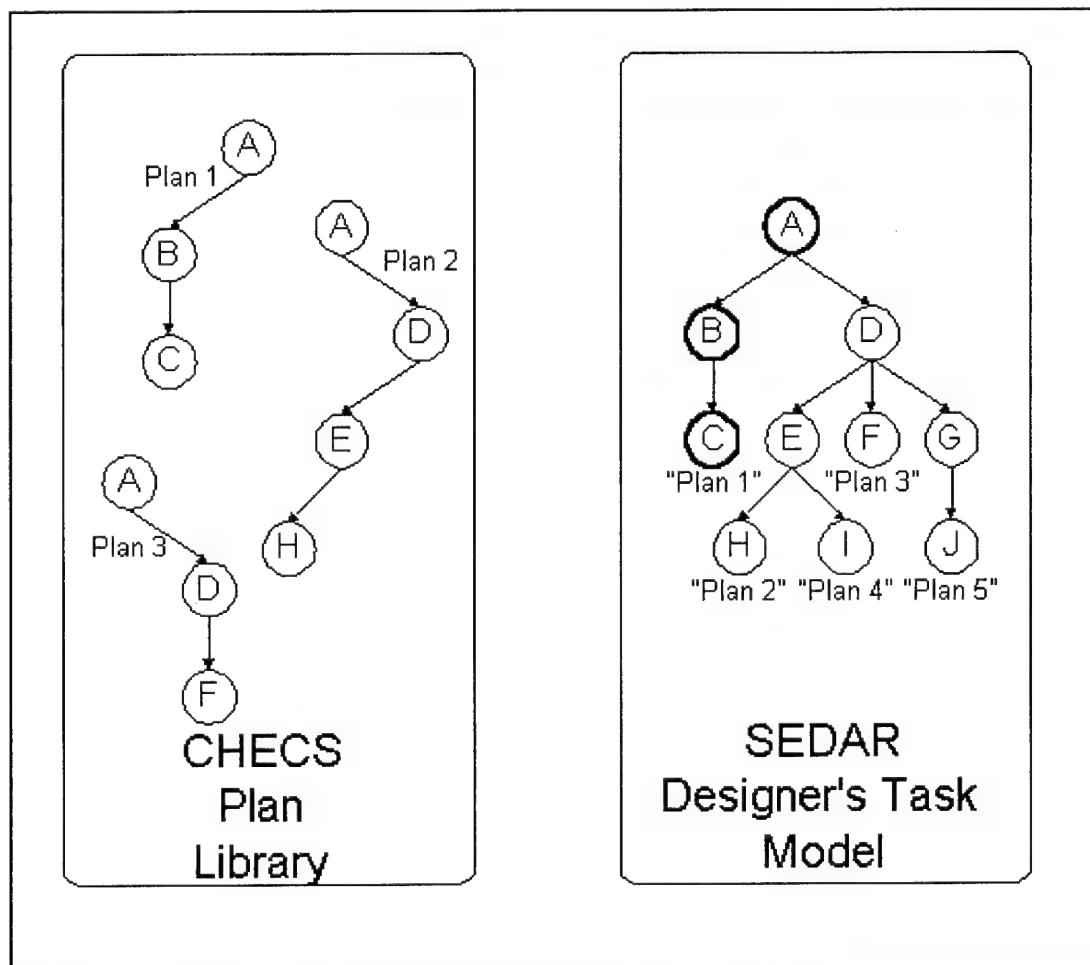


**Figure 22. The CHECS Plan Library and SEDAR's Designer's Task Model.**

system's knowledge of the structure of
the artifact being designed. As the user
modifies the existing design, the set of
design object instances and semantic
links change to reflect the altered de-
sign.

***The Requirements and Materials Hier-
archies.*** The Requirements Hierarchy
is a set of goals or functional require-
ments that the roof design must satisfy.
The existence of this hierarchy reflects
the distinction between user task mod-
els (the DTM) and user goal models.
SEDAR's requirements hierarchy is
similar in nature to the user goal model
of HYDRA, which is also used to repre-



**Figure 23. Representing interacting plans with the Designer's Task Model.**

sent the design's functional requirements. However, SEDAR's set of functional
requirements is currently static; unlike HYDRA, which allows dynamic redefinition
of its requirements throughout the design process, SEDAR's set of functional
requirements is derived from the set established by the conceptual constructibility
model for low-slope roof systems described in East et al. [1995]. Each functional
requirement is linked to a set of design codes describing design conditions that
satisfy (or violate) the requirement. These design codes constitute the construct-
ibility agent described later in this section.

The interactions between materials on a roof should be considered also for good roof
design. The Materials Hierarchy contains the various materials used in roofing
systems. Individual roof components inherit not only from their parent object types
but also from a material; for example, a roof deck may be made of steel, wood, or a
type of concrete.

### Critic Agents

In a fully-implemented version of SEDAR, a collection of agents perform design
analysis and evaluation. These agents are called design code critic agents (DCCAs).
Each critic agent represents a different perspective on the design process and con-
tains design code rules that pertain to the agent's engineering discipline or design
subtask. Thus each DCCA performs a small portion of the overall critiquing task.
The only agent currently implemented in SEDAR is the constructibility DCCA. The
constructibility agent contains the design codes addressing roof operability and

constructibility taken from the *NRCA Roofing and Waterproofing Manual for Low-Sloped Roofing* [1985] and other sources of design and review for low-slope and flat roofs. In future versions of SEDAR, the constructibility agent will be one of a collection of agents performing critiques on the design. The variety of agents reflects the interdisciplinary nature of the roof design task. Besides the constructibility DCCA, there might be a structural engineering DCCA, a mechanical engineering DCCA, a materials DCCA, etc. Each of the agents represents a perspective from a different engineering discipline. For example, the structural engineering DCCA would critique the roof framing plan supporting the roof and how loads on the surface of the roof are supported by the underlying deck and framing structure.

Design codes in DCCAs are either object-relation or object-existence codes. Object-relation design codes express desired spatial relationships between roof objects. A violation of an object-relation code is expressed as an interference between two or more existing objects on the design. Object-existence design codes are used in SEDAR's design suggestion capability. A violation of an object-existence code is expressed as a possible object to add to the design.

Every design code in a DCCA consists of three portions: the rule frame, the rule trigger, and the rule condition. Figure 24 shows the representation for a particular design code. The rule frame contains information about the design code, the rule trigger contains a subset of all the firing conditions for the design code, and the rule condition contains the rest of the firing conditions. The design code is split into trigger and condition portions for efficiency purposes.

### The User Interface

The user interface is an augmented computer-assisted design system that allows direct manipulation of both the design and the criticism generated by SEDAR. This part of the system may also be termed the "front-end" of SEDAR—it is the medium through which the human designer and the critiquing system interact. Furthermore, the user interface constitutes a powerful design environment within which the user may compose a design, control the critiquing system, and view the generated critiques. The interface is described in greater detail in Chapter 8. Figure 25 is a screen capture of the SEDAR interface screen, which shows a partially completed roof design and a critique generated by the system. Displayed in Figure 25 is the Action Menu from which the user selects operations to perform on the design. The interface is divided into three windows: the large area containing the top-down view of the roof design is the Design Window. Critiques generated by the system are displayed here. The small window at the upper left corner of the Design Window is the Suggestion Window. Critiques that involve design suggestions use this window in
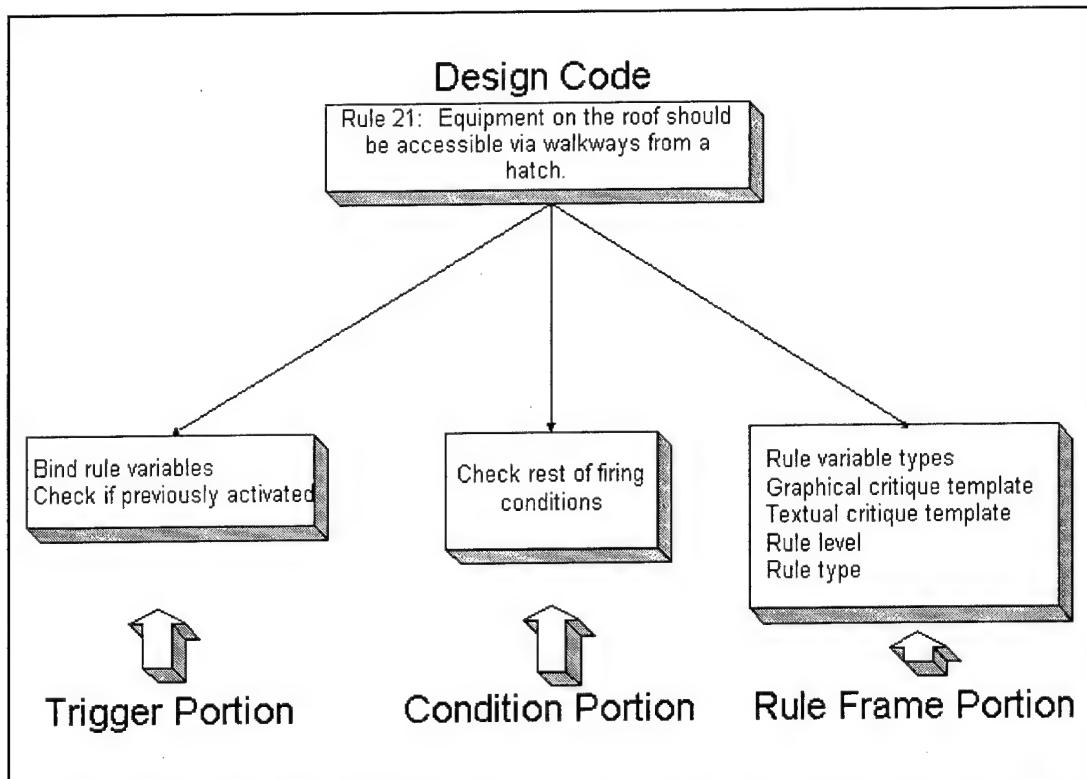
**Figure 24. Design code representation.**
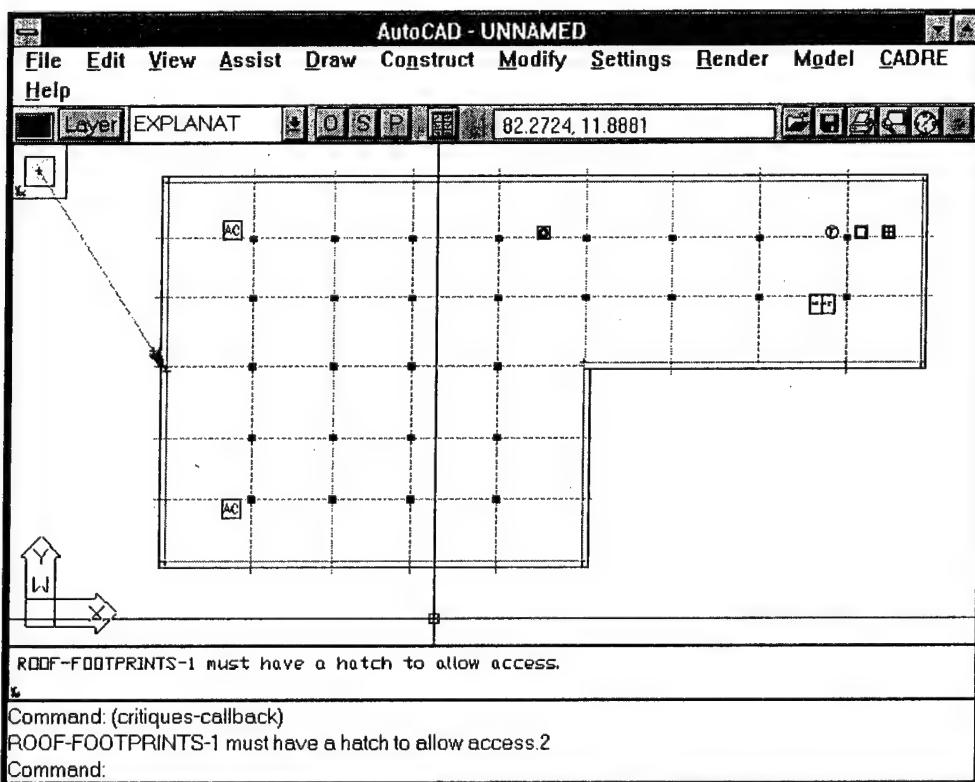


**Figure 25. The SEDAR user interface.**

addition to the Design Window. In Figure 25, the current suggestion is that a hatch be placed on the design to allow access to the roof from below. The suggested hatch object is shown in the Suggestion Window. Finally, the Dialog Window at the bottom of the Design Window displays textual information, including prompts and the textual portions of critiques.

## System Operation: The Iterative Critiquing Cycle

SEDAR uses the iterative critiquing cycle, which forms the framework in which we organize all of SEDAR's actions. The cycle has six stages, shown in Figure 26. Each phase of the cycle is annotated with the components that are involved in its completion. While an overview of each of the stages is presented below, specific stages are presented in greater detail in subsequent chapters. The Update DTM and Design Representation stage has already been partially addressed in this chapter and is described in more detail in this section. The Generate Critiques and Display Critiques stages are critical parts of the overall process that have not been described yet, and will also be discussed in greater detail in this section and later chapters.
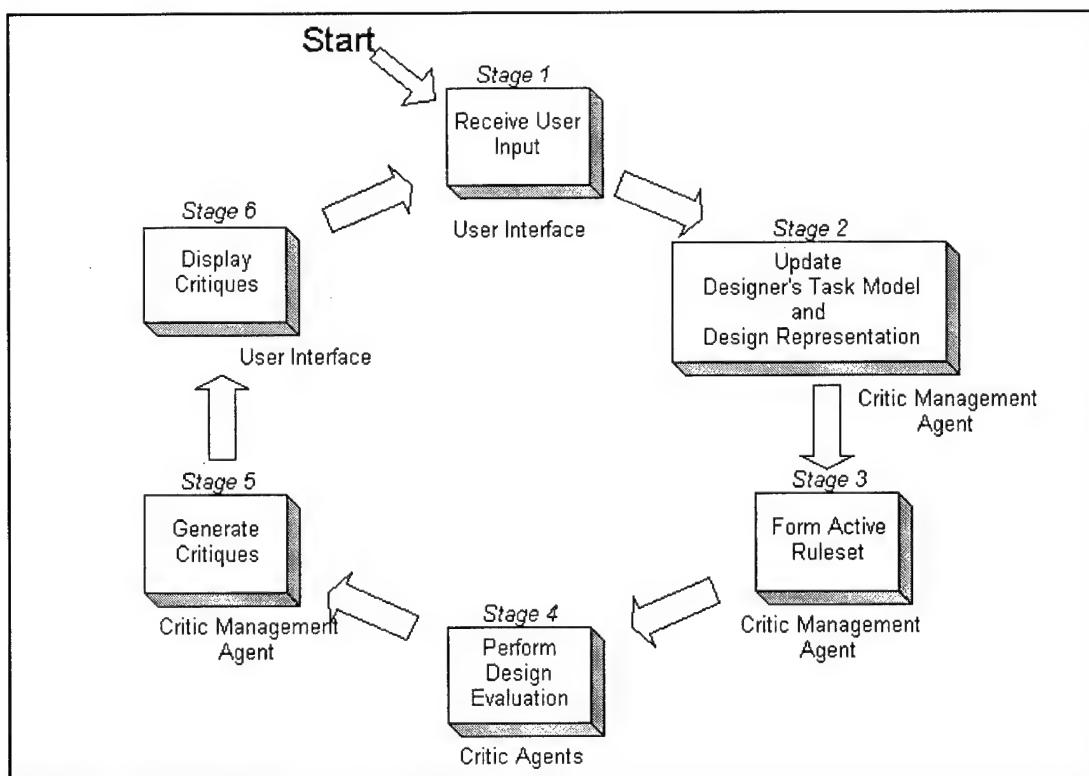


**Figure 26. The iterative critiquing cycle.**

### Stage 1: Receive User Input

The user selects an action to perform, such as adding, moving, deleting, or resizing existing design objects, or selecting goals for review. Depending on the selected action, the interface may query the user for additional information. The interface then sends a message to the critic management agent notifying it of the user's action and providing information that the critic management agent will need.

### Stage 2: Update the Designer's Task Model and the Design Representation

Upon receiving the message from the user interface, the first task of the CMA is to update the DTM. Specifically, the CMA uses the previous DTM activation pattern and the current user action to decide which tasks in the design goal model to make focus or active for the current critiquing session. This method of goal activation allows for greater flexibility in the interaction between the user and the system. For example, some users may like to operate multiple tasks simultaneously. While SEDAR does not actively enforce a particular ordering of satisfaction of its goals, it does have the capability to provide suggestions as to which tasks should be dealt with before or concurrently with the current set of tasks.

The second task for the CMA is to modify the design representation according to the user action. For example, the CMA may make a "temporary" object or a "real" object. If a real object is instantiated on the design representation, additional semantic links may also be created at this time to link the new design object to the previously existing objects.

Finally, the critiquing strategy is selected. Depending on the user's actions, the CMA selects from the error prevention, error correction, and design review critiquing strategies. The method of selection is static in nature and is described in greater detail in Chapter 8.

### Stage 3: Forming the Active Rulesets

During this stage, the set of design codes to be applied for the current critiquing cycle is created. Since the current implementation contains a single DCCA, all design codes are taken from the constructibility agent. Only the rules linked to tasks with focus and active activations in the DTM are included in this set. If there were additional DCCAs available, the set of design codes for each DCCA would then be combined into a single large set of design codes.

The CMA may then modify the rules in the active ruleset, depending on the critiquing strategy. This modification is done to focus the activity of the next stage on relevant objects, and to improve efficiency. This operation is described in detail in Chapter 8.

### Stage 4: Perform the Design Evaluation

The active set of design rules is then applied to the existing design on the blackboard. Each design code rule is a condition-action rule taken from a published handbook of low-slope roofing specifications [NRCA 1985]. If the preconditions of a design code rule match a set of features in the design representation, a design code violation is specified with respect to those features. In every critiquing cycle, only a subset of the knowledge base of rules is applied to the design. This improves the efficiency of the design evaluation stage and, more importantly, ensures that the set of critiques and suggestions provided by the system is appropriate given the state of the design and is relevant to the user's current focus.

### Stage 5: Generate Critiques

In this stage, the violation data from the previous stage are collected by the CMA and are used to generate the critiques seen by the user. An overview of this important element of the process is described here—Chapter 8 describes the transformation process in greater detail.

Critiques have separate graphical and textual portions. The CMA uses design-code specific information to create a graphical critique component in a graphical language understood by the user interface. In particular, the violation data is used to instantiate unbound variables in a stored graphical component template. The textual component generation process follows the graphical component generation. An explanation template containing unbound variables is instantiated with the violation data.

During this stage the critiques are also arranged in order of display to the user. The DTM plays an important role here—the critiques most relevant to the current goals of the user have greater priority over the rest of the critiques, which are arranged according to a serialization of the before-task partial goal ordering.

### Stage 6: Display Critiques

Depending on the critiquing strategy, the user interface may show the graphical/ textual critiques immediately or by user request. The error prevention strategy

displays all of the generated critiques on the drawing without user prompting. The error correction and design review strategies, however, simply display a notification to the user that critiques were found.

After the final Display Critiques stage, the system loops back to Stage 1 and waits for a user action on the design. The process terminates when the user exits from SEDAR.

## Critiquing Strategies

SEDAR currently supports three distinct critiquing strategies: error prevention, error correction, and design review. The strategies differ along many of the characteristics discussed in Chapter 2 of this report—timing, intrusiveness, and intention being the most important issues. The error prevention and error correction strategies are incremental and active in nature and are situated at the design-object activity level of user actions. The basic unit of user activity that encapsulates the iterative critiquing cycle described in the previous section is design object selection and placement. In other words, these critiquing strategies are applied when individual objects are added to the design. This feature has both advantages and disadvantages, which are described below and in Chapter 10. The design review critiquing strategy is a passive, batch-processing critic similar in nature to early critiquing systems.

Each critiquing strategy is presented in more detail below and is illustrated in Chapter 8's example. Discussed in particular is how each strategy fits into the general framework of the critiquing interaction model presented in Chapter 1, and its implementation in the iterative critiquing cycle.

### *Error Prevention Strategy*

Preventive strategies attempt to steer users away from anticipated error patterns before they have the chance to commit them. Since such strategies must take an aggressive approach to advice giving, they are active and either before-task or during-task critics. The difficulty with preventive measures is that it is difficult to decide what type of advice to give—in any situation, there are many possible errors that the user could make. The challenge is to provide useful and appropriate advice to the user in a clear, nonoverwhelming way. Given this set of conditions, few expert critiquing systems employ before-task, active error prevention strategies. An example of a system that offers preventative advice is JANUS, which offers examples of good kitchen designs to the user in response to critiques about the current

design. Displaying good kitchen designs to the user may influence them to follow the example in their own work, thereby avoiding other potential pitfalls that are eliminated by the good design.

The error prevention strategy of SEDAR is of a more principled and direct nature. The strategy shows the user all of the "off-limits" (constraint) areas on the design as defined by the design codes before the object is placed on the design. It is an incremental and active strategy. As was noted in the introduction to this section, the error prevention strategy operates at the design-object activity level. SEDAR divides the user's object-placement activity into two portions (or acts). The first act is object selection, where the user selects the desired design object from a set of design objects. The second act is the actual location and placement of the object on the design. The error prevention strategy occurs after the object selection act and before the object placement act. After a design object is selected and before the user is allowed to place the object on the design, the constraint areas are shown on the design. The constraint areas provide clear cues as to where **not** to place the new object, and only the areas that pertain to the selected object's type (and focus of attention of the user) are shown.

The user may turn this strategy on or off in the user interface. When on, the strategy runs incrementally as described above. When the strategy is off, no advice is given after the act of object selection.

### Error Correction Strategy

The error correction strategy complements the error prevention strategy. Instead of being preventive in nature, this strategy checks a newly placed object for violations according to the design codes in the constructibility agent. The strategy also operates at the design-object activity level, and is invoked immediately after the object placement act. Its purpose is to perform a recheck of the existing drawing against relevant design rules and to propose simple design suggestion cues. Unlike the error prevention strategy, the error correction strategy does not display the generated critiques immediately on screen. Instead, it displays a message stating that violations of design codes were found. The user may choose to ignore this message and to continue on with the design. Thus the error correction strategy is less intrusive than the error prevention strategy.

SEDAR's error correction strategy is most like that of JANUS and its successor, HYDRA. HYDRA's generic critics, which check for desirable spatial relationships among kitchen design objects, are very similar to SEDAR's error correction strategy. The principal difference is in the content of the critiques and how they are displayed.

HYDRA's problem-specific critiques are only textual; general, nonproblem-specific design examples are shown as a part of the hypertext argumentation system. SEDAR generates both problem-specific textual and graphical critiques. The graphical critiques are shown directly on the design. For example, two air-handling units too close together are highlighted and encircled on the design. Additionally, critiques in HYDRA are shown automatically when generated and thus are more intrusive than SEDAR's correction strategy. Like the error prevention strategy, the error correction strategy may also be turned **on** or **off**.

### Design Review Strategy

The final critiquing strategy can be viewed as a passive, after-task, batch-processing critic. Such critics have been integral components of critiquing systems since CRITTER. In SEDAR, the design review strategy serves as an interactive review tool that may be used by either the designer or reviewer. A designer might choose to activate the design review strategy at various points in the design process (e.g., before a review is conducted) to discover errors. A reviewer might use this tool on a design to be reviewed to guarantee that the design satisfies basic constructibility requirements. The benefit of the passive, batch-processing critic is that the user has complete control over when and how the critic is activated, thus eliminating the issues of timing and intrusiveness so critical to active, incremental critics. The problem with such strategies is that these may be applied too late to prevent or to correct errors made earlier, which may lead to some redesign if the problem is severe.

Automated reviews on the existing design are performed by selecting subsets of tasks in the DTM for review. As discussed in Chapter 1, one primary problem of the current design/review process is that the prohibitive cost of performing design reviews has a detrimental effect both on the thoroughness and frequency of the reviews. The design review strategy is a means of better integrating the design and the review components of the design/review process. As an automated review system, the strategy is both more complete and faster than traditional design reviews. Furthermore, reviews may be conducted at any point in the design process. The combination of these two factors leads to more frequent and more complete design reviews.

# 7  Knowledge Representation

Critiquing is a knowledge-intensive process. The representation employed to encode knowledge is a significant factor in how effectively the knowledge is exploited in the system. This chapter describes and motivates the various forms of knowledge representation used in SEDAR. The various components of the system (initially described in Chapter 6) are discussed in greater detail in this chapter. The lone exception is the DTM, which was described in detail in Chapter 6.

The approach used in SEDAR is to use a variety of representational methods. In low-slope roofing literature, much information about the domain is hierarchically organized, thus frame hierarchies were used to represent this knowledge. The design codes found in the NRCA waterproofing manual were fundamentally condition-action rules, and lent themselves naturally to the rule representation described later in this chapter. Besides representational and inferential efficiency, another motivation for using a varied representation is user understandability. User understandability is important because critiquing systems should not be "black boxes," inscrutable to the user. Users will have more confidence in their acceptance or rejection of the system's advice if they can better understand the knowledge and process used to generate the advice.

## Design Representation

This section describes the design representation used in SEDAR in greater detail. Also, specific simplifying assumptions about the roof domain are made in the design representation and are clarified in this section. The representation of the design consists of a set of instances of design objects and a set of instances of semantic links between the design objects. Design objects represent physical components of the roof, such as the deck, a vapor retarder, and an air-conditioning unit. Users may place design objects on the roof, delete them, move them around, or resize them. The roofing design is completed when all the necessary design objects have been selected and laid out on the roof. Semantic links represent relations between design objects. For example, an air-conditioning unit may be *supported-by* a wooden curb, or a walkway may be *adjacent-to* a fume hood. SEDAR automatically adds semantic links to its design representation on an object-by-object basis.

### Design Objects

The generalized form of each design object is kept as a frame in working memory. Each type of design object has its own frame representation. The design object frames are organized in a *part-of* hierarchy (Figure 27*). The root of the tree is the abstract physical-system-components object. All the nonleaf nodes of the hierarchy are used as shell classes and thus are noninstantiable (e.g., roof-system-components, drainage-components). The leaves of the hierarchy are the instantiable design objects (e.g., roof-drains, ac-units-curbed, and attic vents). Each design object inherits from its parent in the design object hierarchy, from a set of material frames, and from a shape frame. The latter two types of frames will be discussed later in this chapter.

The design object frames have slots that describe and structure the attributes associated with the type of design object represented by the frame. When the user selects and places a design object on the drawing, an instance of the generalized design object is made and its slot values filled. Figure 28 is an example of an instantiated design object.

***Geometric Representation and Reasoning***. Every design object has a set of slots describing the intrinsic shape of the object. SEDAR supports two types of shapes: circles and rectangular-compositions. In Figure 28, the shape-type, coordinate-info, horizontal-borders, vertical-borders, and extent slots are used to represent the object's geometric information. SEDAR uses a two-dimensional representation for the low-slope roof layout problem. Both the roof and the objects on the roof are seen from a "top-down-view," as is shown in Figure 29.

Every design object has one of two possible intrinsic shapes: circle or rectangular composition. The *circle* is specified by a center point and a radius. The rectangular composition is any combination of adjacent rectangles. Figure 30 shows examples of both circle and rectangular composition shapes. The simplification of object shape representation to the two distinct types described above is appropriate for two reasons. First, the simplification significantly decreases the difficulty of modeling objects and creating geometric reasoning methods. Second, the simplification was based on a study of low-slope roof objects. It was determined that the two shapes were sufficient to represent a large range of possible roof objects. One limitation of SEDAR's geometric representation is that objects must be arranged along rectilinear lines (parallel to the x- and y- axes). This restriction was decided on after an examination of roof designs showed that nonrectilinear arrangements are rare in the

---

* Figures appear at the end of the chapter.

roofing domain. However, as the system expands into domains other than low-slope roofing, the need to represent nonrectilinear arrangements will clearly become necessary.

Besides limiting the computation of geometric relations to an as-needed basis and storing the results of computations for reuse, researchers tried to make the LISP-based geometric reasoning methods more efficient by using a two-layered reasoning process. The first layer is meant to filter out "easy" rejection cases and to eliminate them from further (potentially expensive) computations. The technique that was used was to create extents, which are minimum bounding rectangles around object shapes. Figure 31 shows a set of rectangular-composition shapes and their extents.

To illustrate the use of extents, Figure 32 depicts the action of the complete-overlap LISP function on two different scenarios. The first scenario has two rectangular compositions, RC-1 and RC-2. Figure 32a shows RC-1 completely overlapping RC-2. The initial check of the extents shows that the extent of RC-1 overlaps the extent of RC-2 completely. Thus a more involved check using the rest of the geometric information stored in the rectangular-composition objects is performed. In Figure 32c, however, RC-2 is now not completely overlapped by RC-1, so the initial check of the extents fails because part of RC-2's extent is outside RC-1's extent. So the relation fails immediately, and no further checking is required. The second scenario involves a rectangular composition, RC-3, and a circle, C1. In Figure 32b, C1's extent is completely overlapped by RC-3's extent, thus leading to the more expensive check. The interesting case is that illustrated by Figure 32d. Here, C1's extent is fully within RC-3's extent, but RC-3 clearly does not completely overlap C1. Although the initial test on extents is successful, it is the more expensive check on the rest of the geometric information that causes the relation to fail.

Improving the efficiency of the reasoning routines themselves was essential because of the nature of design checking performed by SEDAR. First, design checks are meant to be exhaustive with respect to each active task. For example, when a new design object is added to the design, all possible, relevant interactions are investigated. When a new air-conditioning unit is added to the design, the complete-overlap relation is checked against all the drains, sumps, fume hoods, other air-conditioners, etc. on the design. Since most of the objects will not satisfy the complete-overlap relation, filtering by reasoning about extents is both a justifiable and efficient means of reducing computational complexity.

*Other Design Object Slots*. A class of design objects may also have its own special slots. For example, when dealing with plywood roofing decks, a grade of wood must be specified, or when working with steel roofing decks, both the gauge of the steel

and the flute width must be specified. The air-conditioning object in Figure 28 has one special slot, orientation, that can take one of four values: N, S, E, or W. This slot specifies the compass direction that the air-conditioning unit faces.

The designer may change some of the slot values of a design object directly. In SEDAR, the set of slots that the designer may directly alter is contained in the user-modifiable-slots. For the air-conditioning unit, the only user-modifiable slot is the orientation slot. The designer may also alter the values of slots indirectly. For example, the geometric information slots described in the previous section are updated by SEDAR whenever the designer moves or resizes the design object. This removes the burden on the designer of entering coordinate values by hand and of ensuring consistency between the geometric information slots.

The special *activate-when-created-ruleset* and *activate-when-created-functions* slots are used when an object is created. The *activate-when-created-ruleset* slot contains the name of a set of forward-chaining rules that are fired when the object is created. The *activate-when-created-functions* slot lists LISP functions to call when the object is created. Both the rules in the ruleset and the LISP functions make assertions (or semantic links), in the design representation that express a relationship between the newly created object and the other objects on the design. When a designer deletes an object from the drawing, the corresponding design object instance is deleted, as are any semantic links that relate to the design object. Additional functionality is available by using the *activate-when-deleted-ruleset* slot to designate rules to fire upon the deletion of the design object. An example where this ruleset is useful is when drains are deleted. The NRCA specifications call for a minimum number of drains for different roof sizes. A count of the drains on the roof can be maintained by using rules fired when a drain object is created or deleted.

### Semantic Links

Designs in SEDAR are also specified by semantic links amongst the design objects. Semantic links represent inferred relationships between objects. Such relationships are mostly topological in nature, although some higher level semantic links may capture some functional relationships. A hierarchy is implicit among the semantic links. "High-level" semantic links use "low-level" semantic links in their definitions, while "low-level" semantic links are directly implemented in LISP.

An example of a "low-level" semantic link is complete-overlap, which was shown in Figure 32. *Complete-overlap* is a binary relation; *object-1* is related to *object-2* with the *complete-overlap* relation if the area of *object-2* is completely within *object-1*. An example of a "high-level" semantic link is the *supported-by* relation. Heavy pieces

of roof-mounted equipment must be supported by wooden curbs built into the structure of the roof. High-level semantic links are compositions of low-level semantic links. For example, the *supported-by* relation checks to make sure that the supported object is spatially arranged correctly on its curbing—done via verifying a series of "lower-level" semantic links.

Like design objects, instances of semantic links are created in the design representation to describe the relationship between objects. Unlike design objects, the user has no direct control over semantic link instantiation. Semantic link instances are created in two ways. The first way is to be created when the object is created via the ruleset named in the *activate-when-created-ruleset* slot of the object. The second way is to be created when design rules are fired. Many design rules depend on the existence (or absence) of a type of semantic link between two or more objects. When a link is checked for in the *if*-portion of a design rule, if the link is unknown, it is created and saved for future reference. The primary motivation for this approach was computational efficiency. Each semantic link frame is associated with a LISP function that checks the appropriate relationship between two (or more) objects. For example, the euclidean distance semantic link frame is associated with a LISP function that computes the minimum euclidean distance between two design objects. For even simple, low-level semantic links, the LISP functions may be computationally expensive. In our initial design rule checking strategy, every time the preconditions of a design rule were checked the LISP function was called. This approach proved unsatisfactory, as an unacceptably large amount of time was spent recomputing relationships. Our second approach was based on two principles:

1. Compute relationships only on an as-needed basis.
2. Reuse the results of previous computations as much as possible.

## Requirements Hierarchy

The Requirements Hierarchy shown in Figure 33 is modeled after the requirements established by East et al. [1995] and discussed in Chapters 5 and 6. The functional requirements represent the goals that a roof design must satisfy. Each individual requirement is associated with a set of design codes that address problems with respect to the requirement. Therefore, every design code implemented in SEDAR is associated with both a requirement and a set of design tasks in the DTM. Although the current implementation of SEDAR has no provision for performing reviews using the Requirements Hierarchy (only using the DTM), the ability to do so will be added shortly. While designers may find the DTM natural, reviewers are more likely to find the Requirements Hierarchy more natural from their perspective.

Eventually, a user should be able to conduct reviews (using the design review critiquing strategy) based on either the DTM or the Requirements Hierarchy. The designer's perspective is defined by roof subsystem design tasks (e.g., drainage system layout or flashing design); the reviewer's perspective is defined by roof functions (e.g., waterproofing). Thus alternative hierarchies are necessary to accommodate the two types of users.

## Materials Hierarchy

In roofing design, analysis of the combination of materials used for roof objects is essential; certain combinations of materials may lead to disastrous results. One example of this is using any type of bitumen adhesive on a single-ply EPDM roof. Bitumen tends to degenerate the ethylene/propylene diene monomer, resulting in a significantly weakened roof membrane.

Constituents of the materials hierarchy of SEDAR (Figure 34) are used as parents of certain design objects. For example, a deck may be steel, wood, or a variety of types of concrete. Each specific type of deck has as its parents the general design object decks and the corresponding material (see Figure 35). Currently SEDAR has few codes in its knowledge base dealing with roof materials and their interactions. Future work for SEDAR will include the development of a materials DCCA to evaluate the roof material system.

## Design Codes

The design codes implemented in SEDAR come from a variety of sources. The primary source for the codes was the *NRCA Roofing and Waterproofing Manual for Low-Slope Roofing* [1985]. These types of rules are specification-level. Also, many "common sense" or physical-level rules, are used mostly to check for physical impossibilities, like intersecting two air-conditioning units (Figure 36) or placing design objects like roof drains outside the roof footprint entirely (Figure 37). The final type of rule is preference-level, used to express individual designer's design preferences. One roof designer favors having a small overflow drain close to each roof drain in the roof field to handle excess ponding. While not required by the NRCA specifications, this was a strong preference that the roof designer wanted to pass on to other roof designers. Constraint violations resulting from physical-level, specification-level, and preference-level rules are known as physical, specification, and preference violations, respectively.

Each design code has three parts: a trigger portion, a condition portion, and a rule information portion. The condition-action nature of each design code is captured in the trigger and condition portions, which are themselves expressed in a condition-action form. The trigger portion of the design code is used to check the design for the basic applicability of the design code. This involves checking the design for the correct types of design objects and whether the particular set of design objects had ever been checked before. If the basic applicability conditions are satisfied, the condition portion of the rule is invoked. This portion usually involves the calculation of a relationship between the two objects and is generally more expensive to apply than the design code trigger. If the condition portion is satisfied, a note is made of the violation, and a critique is generated. Figure 38 is an example of a design code and its trigger and condition portions.

Both the trigger and condition portions are expressed as *if-then* rules. The antecedent of the trigger portion is a conjunction of conditions. The first two conditions establish the type of objects (i.e., any type of equipment and a hatch) and bind instantiated design objects to the variables (?e1 and ?e2). The third condition, (not-equal ?e1 ?e2), ensures that ?e1 and ?e2 are not the same object. The last condition of the trigger checks to see if the rule (Rule 21) has been checked previously and found not to be in violation. If it has, then there is no reason to continue with the current rule check. The record of previously checked rules is updated when design objects are moved, resized, or deleted. If a design object has been modified, then the previous rule checks are no longer valid.

The consequent of the trigger portion asserts a message for the condition portion of the design code. In particular, it establishes an identification tag for the rule check and the variable bindings for the check.

The antecedent of the condition portion performs the actual violation check between the objects. In the example, this check is performed in the line (equal (connected? ?e1 ?e2) '()). The connected? relation is implemented as a LISP function, which checks to see if the two design objects are accessible via a sequence of walkways. If the relation fails, the equipment object is not accessible via the hatch and a violation message is created, to be processed in the Generate Critiques phase of the iterative critiquing cycle.

The reason for splitting the trigger and condition portions is discussed in detail in the next chapter. Figure 39 shows the rule frame portion corresponding to the sample design code. This portion contains the variable-object type association list, the rule level, the rule type, and critique generation information.

Whether a rule is physical-level, specification-level, or preference-level determines its precedence when multiple problems with an object are discovered. In general, physical-level rule violations are more serious than specification-level rule violations, which in turn are more serious than preference-level rule violations. So if a particular object like the drain in Figure 37 is found to violate multiple rules, only the violations associated with the highest priority levels are reported to the user. In Figure 37, since the drain is completely off the roof, it is not necessary to mention that it is not at one of the points of highest deflection in the roof field (a specification-level rule violation) and that it does not have an overflow drain next to it (a preference-level rule violation).

Aside from being physical-level, specification-level, or preference-level, each rule may either be an object-relation rule or an object-existence rule. Object-relation rules detect problems between existing objects on the design; object-existence rules make suggestions for adding (or removing) objects to or from the design. The example in the next section will highlight the differences between these types of rules.

The critique generation information from the text, bindable-list, explanation, and violation-action slots is used to create the graphical/textual critiques described in the next chapter. The graphical component of the critique is generated from the contents of the violation-action slot, and the textual component of the critique is generated from the contents of the explanation slot.
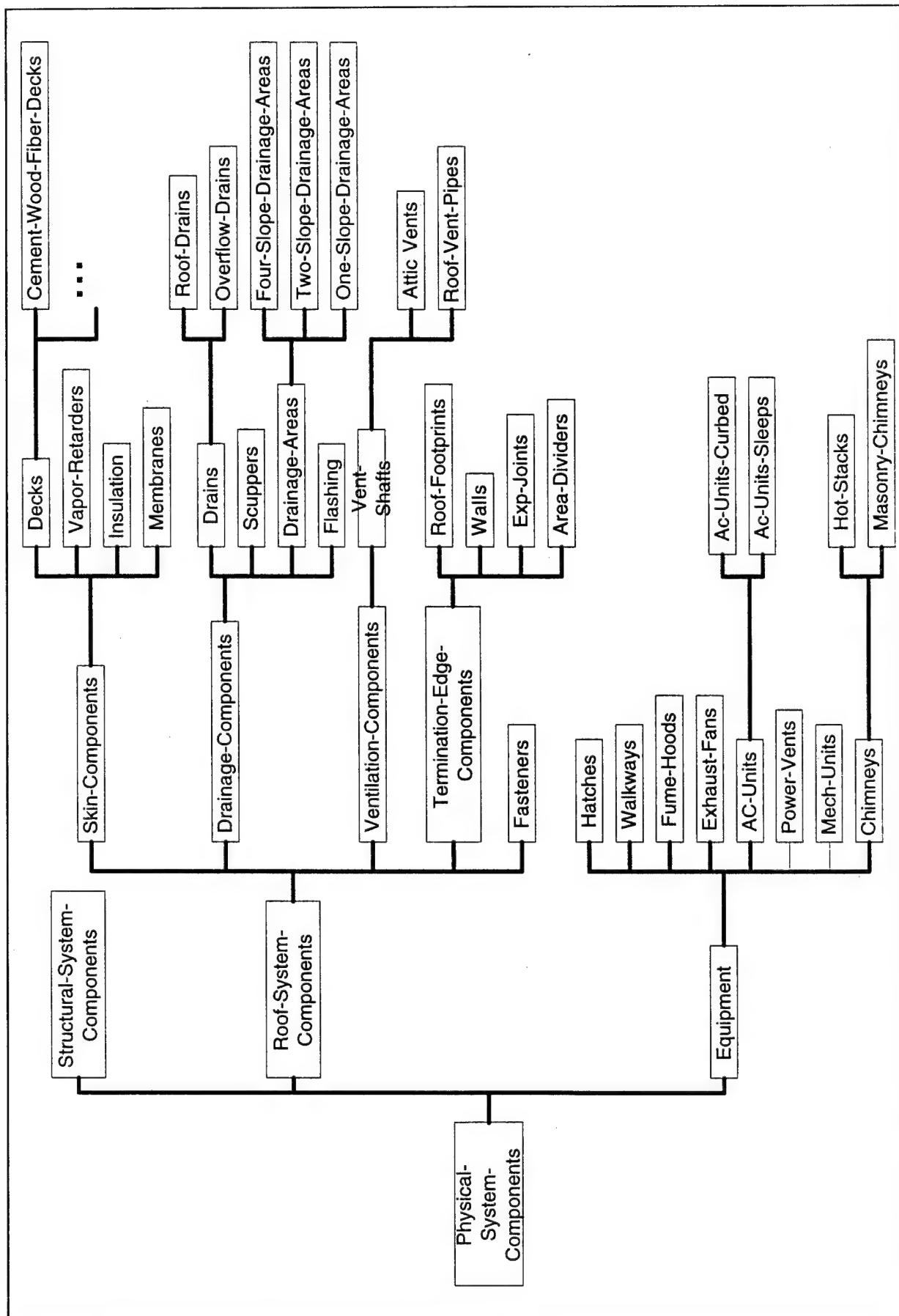
Figure 27. The Design Object Hierarchy.

Instance: AC-UNITS-CURBED-1
Parent: AC-UNITS-CURBED
Slots:

| | |
|---|---|
| Shape-Type | RECTANGULAR-COMPOSITION |
| Coordinate-Info | ((42.0989 81.783) (45.0989 81.783) (45.0989 78.783) (42.0989 78.783)) |
| Horizontal-Borders | ((42.0989 (78.783 81.783)) (45.0989 (78.783 81.783))) |
| Vertical-Borders | ((78.783 (42.0989 45.0989)) (81.783 (42.0989 45.0989))) |
| Extent | ((42.0989 78.783) (45.0989 81.783)) |
| Orientation | N |
| User-Modifiable-Slots | (ORIENTATION) |
| Object-Type | REAL |
| Activate-When-Created-Ruleset | EQUIPMENT-RULES |
| Activate-When-Created-Functions | NIL |
| Activate-When-Deleted-Ruleset | NIL |

**Figure 28. An AC-UNITS-CURBED instance.**
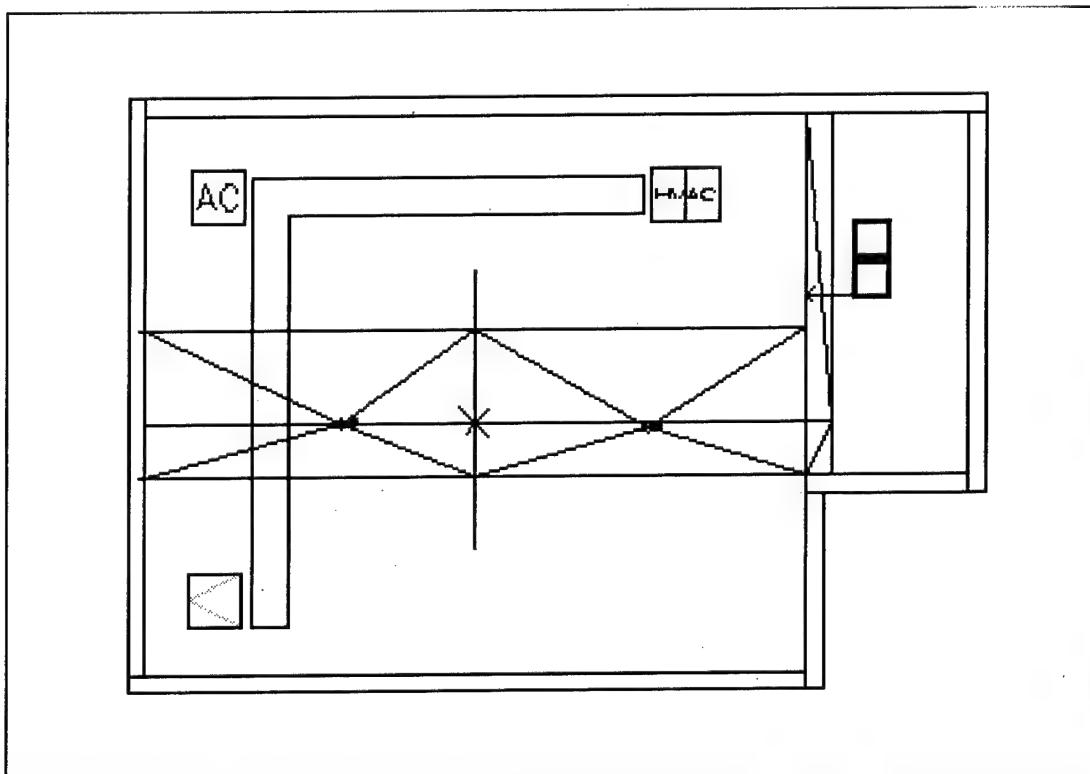


**Figure 29. The "Top-Down" view of the roof.**
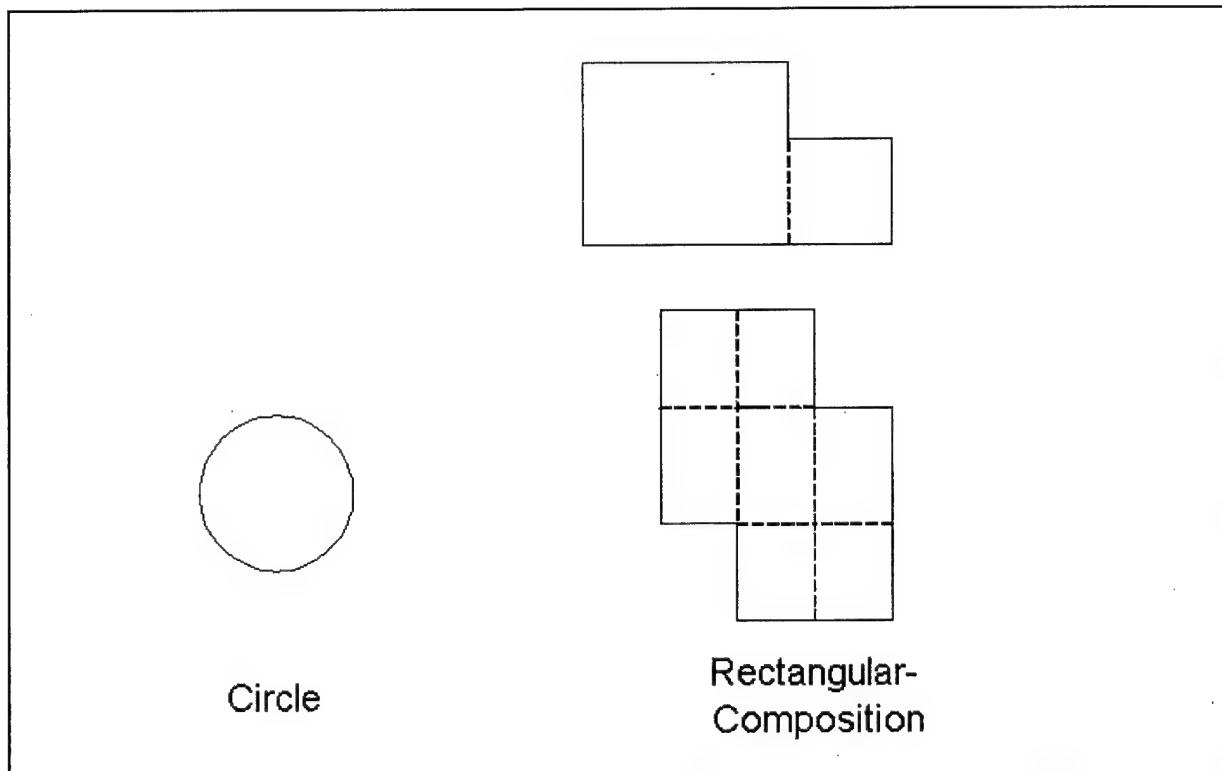
Figure 30.  The circle and rectangular-composition shapes.
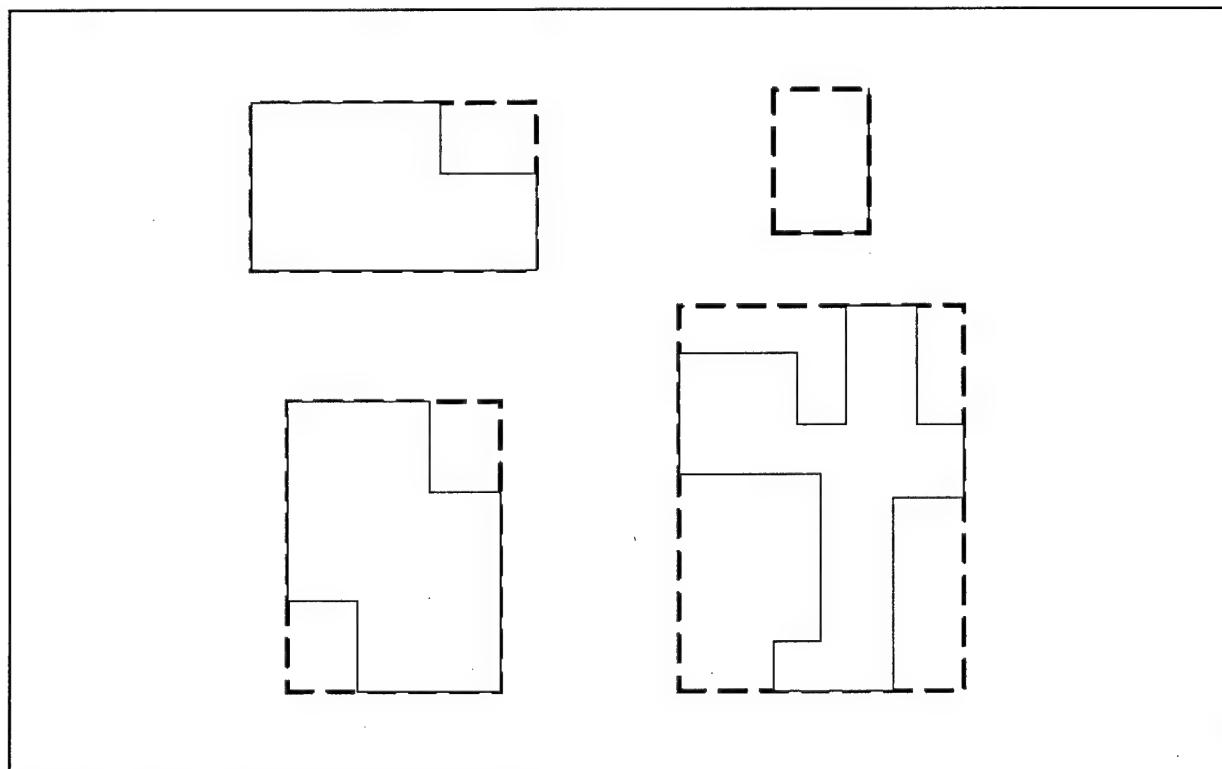


Figure 31.  Extents of rectangular-compositions.
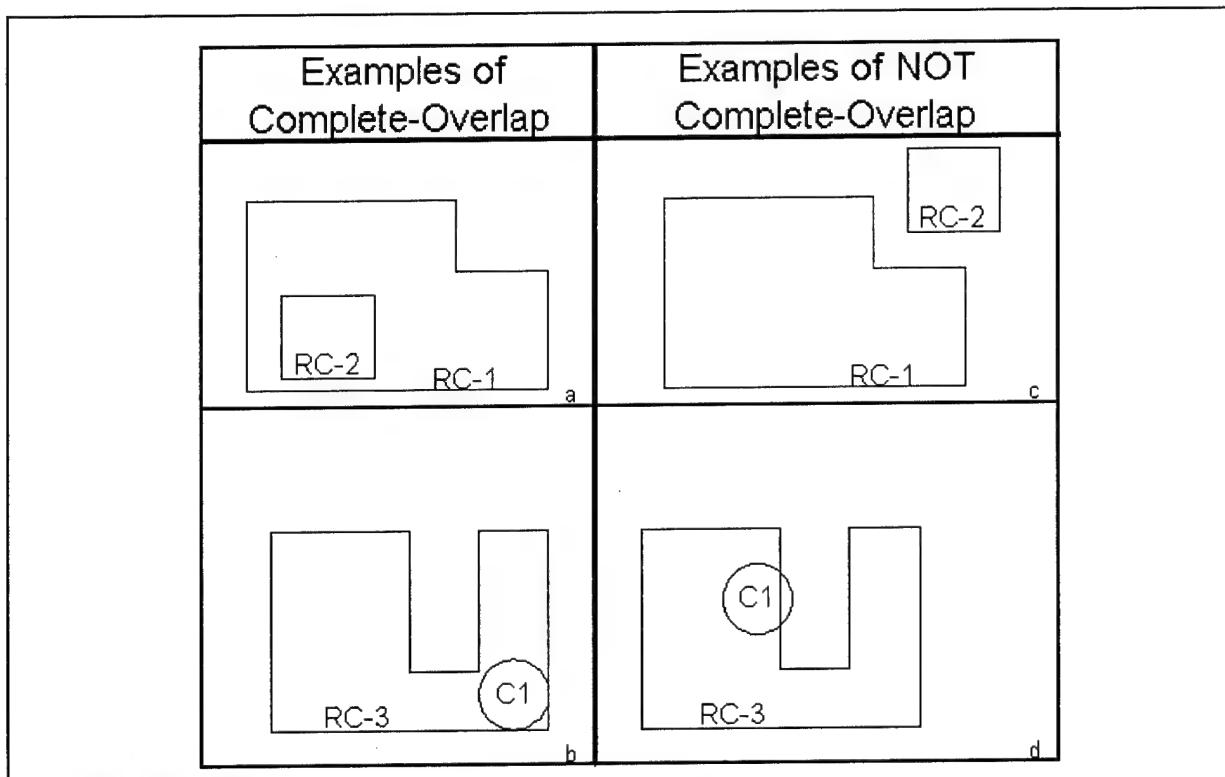
**Figure 32. The complete-overlap relation.**
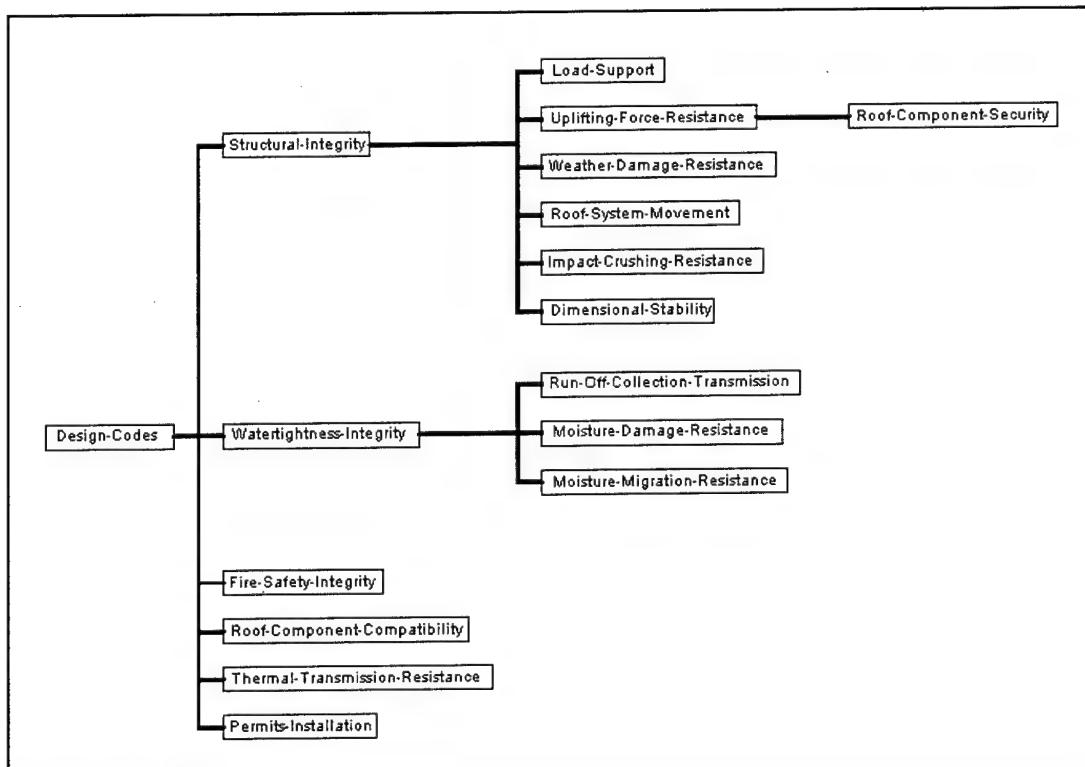


**Figure 33. The Requirements Hierarchy.**
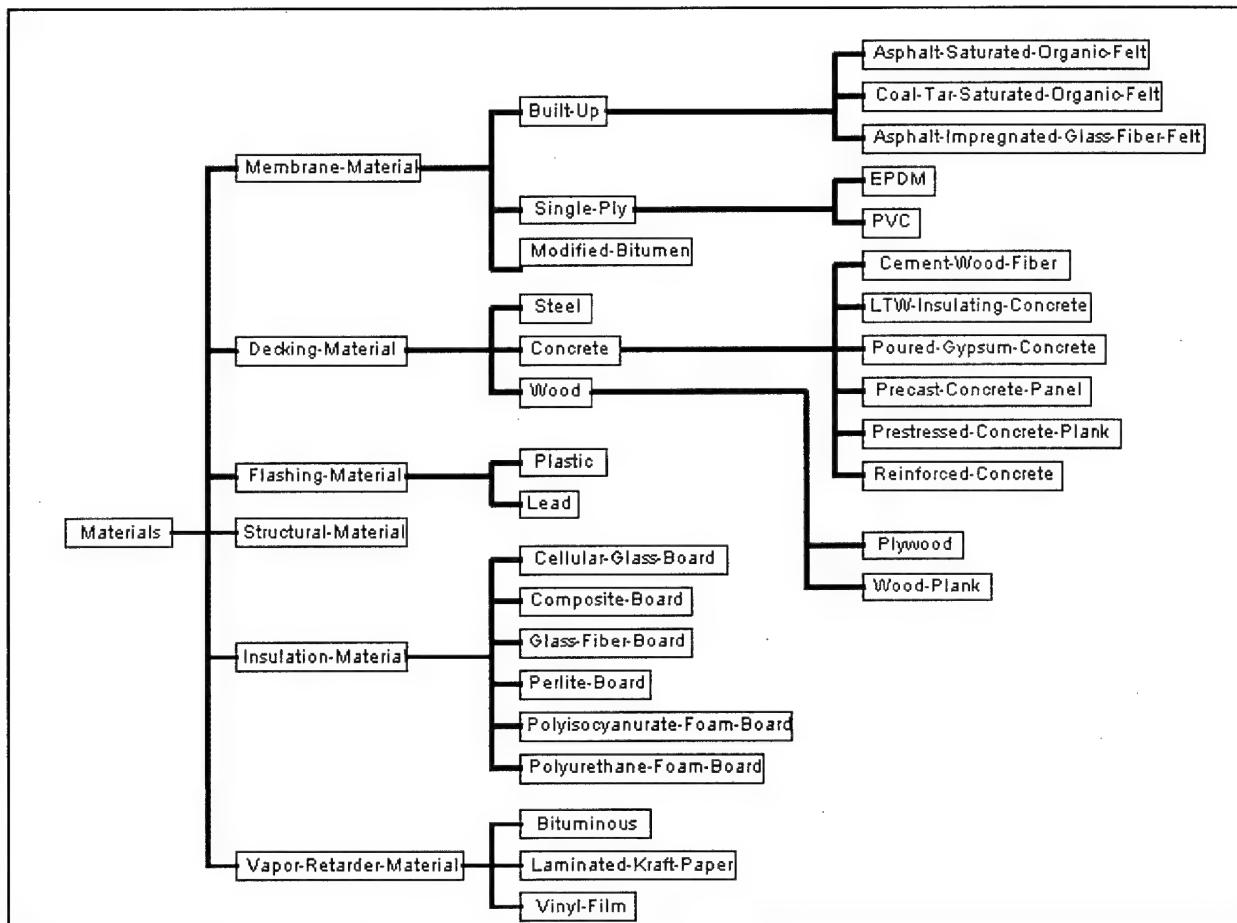
**Figure 34. The Materials Hierarchy.**
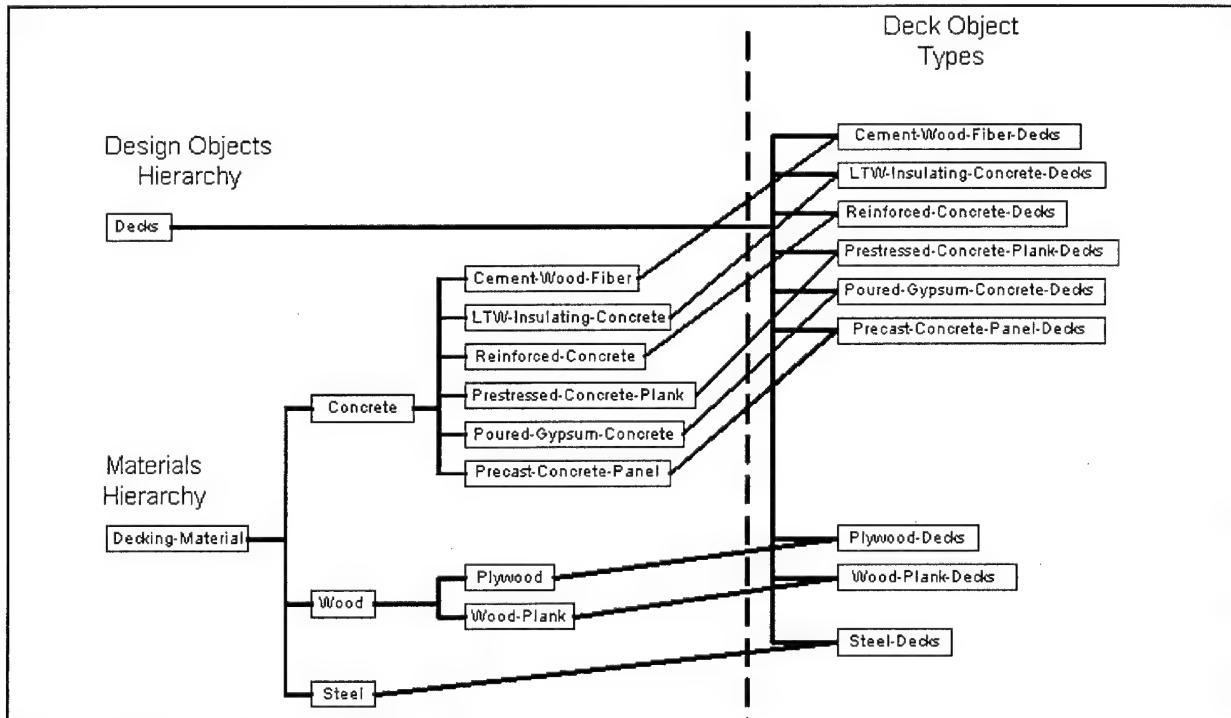


**Figure 35. Parents of deck objects.**
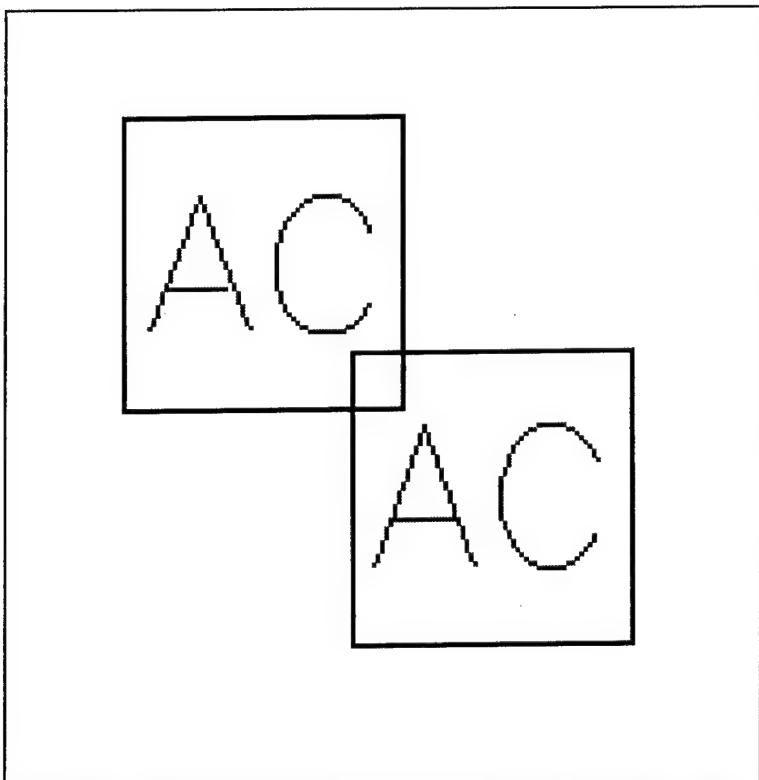
**Figure 36.  Physical-level violation between air-conditioning units.**
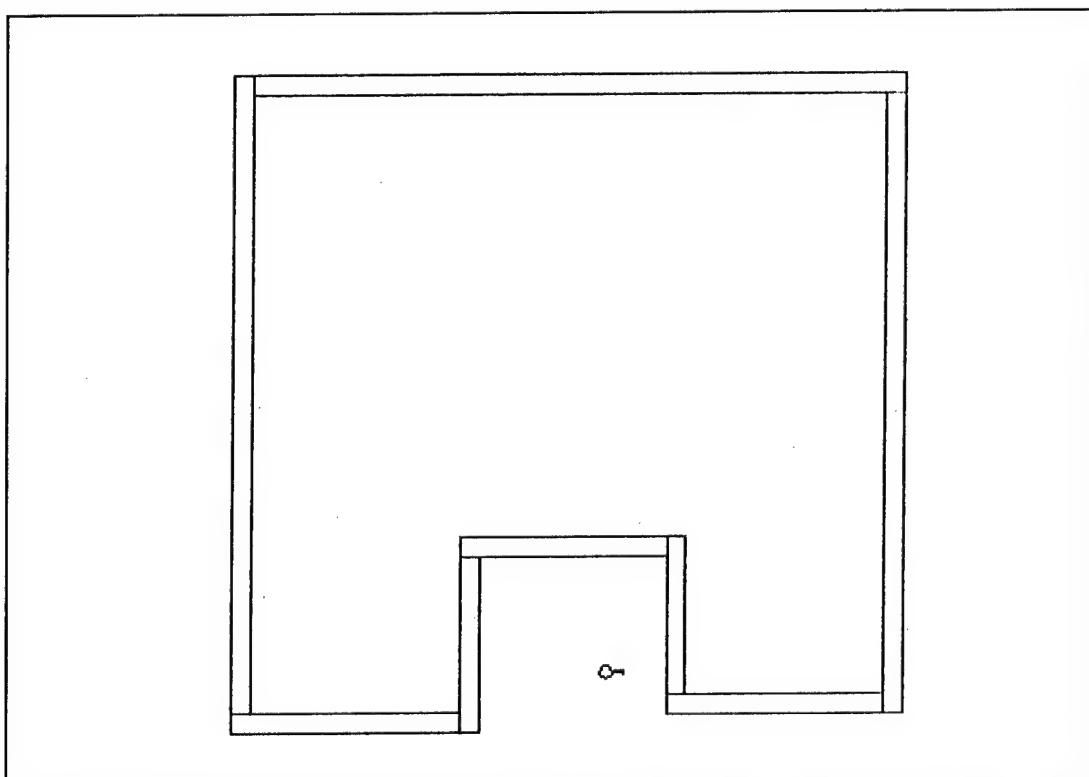


**Figure 37.  Physical-level violation involving a roof drain.**

## Design Code

Rule 21: Equipment on the roof should be
accessible via walkways from a hatch.

## Trigger Portion

```
(define-rule rule21-trigger (:priority 100)
    (instance ?e1 is equipment)
    (instance ?e2 is hatches)
    (not-equal ?e1 ?e2)
    (equal (checked-before-dual 'rule21 (list "?e1" ?e1) (list "?e2" ?e2)) '())
  THEN
    (check-condition ?new-violation-name ?current-query rule21 (("?e1" ?e1) ("?e2" ?e2))))
```

## Condition Portion

```
(define-rule rule21-condition (:priority 0)
    (check-condition ?name ?current-query rule21 ((?t1 ?e1) (?t2 ?e2)))
    (equal (connected? ?e1 ?e2) '())
  THEN
    (violation ?name ?current-query rule21 ((?t1 ?e1) (?t2 ?e2))))
```

**Figure 38. The trigger and condition portions of a design code.**

Instance: RULE21
Parent: DESIGN-CODES
Slots:

| Name | RULE21 |
|---|---|
| Level | SPECIFICATION |
| Rule-Type | OBJECT-EXISTENCE |
| Permanent | T |
| Text | "All equipment should be accessible via walkways from the hatch." |
| Trigger | RULE21-TRIGGER |
| Condition | RULE21-CONDITION |
| Object-Driven | RULE21-INTERACT |
| Bindable-List | ((?E1 EQUIPMENT) (?E2 EQUIPMENT)) |
| Explanation | ("There should be a walkway from " "?e1" " to " "?e2" ".") |
| Violation-Action | (MULTIPLE-DRAW (DRAW-BOUNDARY-AREA "?e1" UNKNOWN INTERIOR 0) (DRAW-BOUNDARY-AREA "?e2" RECTANGULAR-COMPOSITION INTERIOR 0)) |

**Figure 39. The rule frame.**

# 8  Examples of System Operation

This chapter gives a concrete example of how the ideas presented in this report have been implemented in SEDAR. Through a series of screen captures, the actions of a roof designer on a partially complete roof plan in the SEDAR environment are displayed and the system's responses explained. In particular, these examples illustrate how the critiquing strategies—error prevention, error correction, and design review—are related in the current implementation. While previous chapters described the system in a static fashion, this example demonstrates how the various system pieces work together to provide salient, coherent feedback to the system user.

The example begins with the partially complete roof design shown in Figure 40.\* In this flat roof, the column lines (criss-crossing the roof field) are shown as dotted lines. At each intersection of perpendicular column lines, a black square indicates a load-bearing column. The parapet wall surrounding the roof field is also load-bearing in this case. The cursor is composed of the solid-line crosshairs in the bottom right of the drawing. The roof is in the intermediate stage of design—the designer has already placed several objects in the roof field. On the left of the roof are two air-handling units. A power vent is placed at the upper center of the roof, and a chimney, exhaust fan, and HVAC unit are at the right of the roof. The last action taken by the designer was to place one of the air-handling units on the design. The initial activation pattern of the DTM is shown in Figure 41. The air-handling-layout task thus has a focus activation, and the tasks related to the air-handling-layout task by *interferes-with* relations are active. Formerly focus tasks are also active. This activation pattern is the same as the one shown in Figure 21.

### Action 1:  Placing a Masonry Chimney on the Roof Design

Before moving on to address other roof subsystems, the designer wants to complete the layout of roof-mounted equipment and other major penetrations on the design. One of the requirements for the building is that two masonry chimneys be provided

---

\* Figures appear at the end of the chapter.

in the same vicinity on the roof design. To achieve this, the designer clicks the right mouse button in the drawing area and selects the New Object... option from the action menu that appears (Figure 42). Besides adding new objects, the designer may also choose to modify an existing object, look at the Goals... or Critiques... windows, or to save or open a stored roof layout. The Goals... and Critiques... windows are described in detail later in this example.

SEDAR then responds with the object palette shown in Figure 43. After using the scrollbar on the right of the object list to find a masonry chimney, the designer clicks on the OK button. The available objects are the set of instantiable (leaf) objects in the Design Object Hierarchy. The act of selecting an object from the object palette signals the end of the first stage of the iterative critiquing cycle.

### An Application of the Error Prevention Strategy

During the second stage of the cycle, the CMA decides which critiquing strategy to use for the rest of the cycle. The decision to use the error prevention strategy is made when the CMA learns that the designer has selected an object. Since the designer has not yet placed the object on the design, the error prevention critic may be able to provide some useful information to the designer regarding where to place the new object. Also taking place during the second stage is the updating of both the DTM and the design representation. The activations of the tasks in the DTM are updated according to their prior state and the recent masonry chimney design object selection. The new activation pattern of the DTM is shown in Figure 44. The chimney-layout task now has a focus activation, and the activation of the air-handler-layout task has been demoted to active.

In the design representation, a temporary, incompletely specified masonry chimney object is created. No semantic links are created. In the third stage of the iterative critiquing cycle, the set of design codes to be applied to the design is formed. The set of design codes is formed from the union of the individual sets of design codes associated with each focus and active task in the DTM. For the error prevention strategy, only the trigger portions are included in the rule set. The condition portions are not included because their antecedents require geometric information that is currently not known about the object. The trigger portions include only object type checking predicates in their antecedents—thus applying only the trigger portions will result in all possible interactions between the new object and the existing objects on the design. This is useful because the error prevention strategy seeks to show all the constrained areas on the existing design.

After determining which design codes to apply, SEDAR moves into the fourth stage of the cycle, and the set of design codes are fired on existing design. This process results in "violations," or more appropriately, constraint area notifications that specify off-limits areas for the new chimney. During the fifth stage of the cycle, these constraint areas are formed into a set of graphical and textual instructions for the user interface. Each constraint area has a set of bound variables. The graphical critique template is taken from the associated design-code's rule-frame and is instantiated with the set of bound variables for the constraint area. For the error prevention critic, the textual portion of the constraint area is not created. Also, since the location of the selected object is not set, only half of the graphical portion of the critique, that involving the object on the existing design causing the constraint area, is generated and displayed.

The constraints are then ordered using the activation pattern of the DTM. For example, a constraint resulting from a design code associated with the chimney-layout task would be ordered before a constraint resulting from a design code associated with the air-handler-layout task, because the chimney-layout task has a focus activation and the air-handler-layout task is active.

Figure 45 shows the results of the sixth stage of the iterative critiquing cycle. Since the error prevention strategy was applied, all of the constraint area notifications are displayed on the drawing. In the figure, these notifications are shown as hatched areas. Each of the existing pieces of mechanical equipment are shown encompassed in hatched regions, which are the result of design codes specifying minimum distances between objects. Hatching is also around the perimeter of the roof field. This hatching was created by design codes specifying a minimum distance between roof-mounted equipment and the roof's edges. The collection of all these constraint areas shows the designer where **not** to place the chimney on the roof.

The display of the constraint areas concludes the sixth stage of the cycle, and again SEDAR waits for an action by the designer.

### *An Application of the Error Correction Strategy*

The designer then places the new chimney on the design close to the existing chimney. Suppose that the new chimney is too close to the existing chimney, thereby violating one of the design codes in the knowledge base. Placing the object by moving the AutoCAD crosshairs to the desired location and clicking the left mouse button signals the end of the first stage of the critiquing cycle. The action of actually placing an object causes the CMA to activate the error correction strategy. The DTM is **not** updated in the second stage, because no changes in activations

would take place from the previous user action (that of selecting an object). The temporary object in the design representation is updated with the now available geometric information. Furthermore, the system now creates the appropriate semantic links between the other design objects and the new object.

During the third stage, both the trigger and condition portions of the rules are included in the rule set. So the error correction strategy finds actual violations of design code, rather than the possible ones in the error prevention strategy. The fourth stage differs little from before—the rule set is applied to the existing design, which creates rule violations. In the fifth stage, both the textual and complete graphical portions of each violation are generated, ordered according to the DTM activation pattern, and returned to the user interface.

The full critiques are generated because the user may browse through the critiques and examine their explanations. Instead of actively displaying the critiques on the design, the user interface notifies the user that violations have been found (Figure 46), and a textual message shows in the dialog window of the interface.

To view the critiques, the designer selects Critiques... from the action menu. This displays the critiques in the Violations window shown in Figure 47. The Violations window separates the rule violations into the three levels of design codes: physical, specification, and preference. In this case, two separate specification-level violations were found; one relating to RULE6 and another relating to RULE12. The text corresponding to the currently selected violation is displayed beneath the three boxes. The specification violation stemming from RULE6 is that the two masonry chimneys must be at least 1 ft apart. Clicking the View button causes the interface to show the graphical portion of the critique (Figure 48).

The graphical critique for this violation consists of displaying two constraint areas around the chimneys, marking out boundary areas 1 ft outside the respective chimneys. The overlap of one chimney's constraint area and the other chimney expresses the constraint violation. Furthermore, a red outline box encompasses the two objects and their constraint areas. The intent of the box is to both attract the designer's attention and to focus his attention on the objects and constraint areas. Figure 49 is a closer view of the constraint violation displayed in Figure 48.

The designer then decides to move the newly placed masonry chimney to avoid the specification violation. Selecting Move Object from the action menu allows the designer to click on the chimney and to place it further away from the original chimney. The error correction critic checks the drawing again for violations, and

this time RULE6 does not cause a violation. Figure 50 shows the designer's new location for the chimney.

### An Application of the Design Review Strategy

After correctly placing the chimney on the roof design, the designer believes that the functional requirements for mechanical equipment on the building have been satisfied. To test this hypothesis, the designer selects Goals... from the action menu, which allows him to select one of the tasks of the DTM. In this case, the designer elects to review the roof-mounted equipment subsystem and chooses the equipment-layout task (Figure 51).

Despite the differences in the first stage of the critiquing cycle, stages two through six of the design review strategy are very similar to those of the error correction strategy. A rule set is formed with the trigger and condition portions of the design codes affiliated with the selected review goal. The rule set is fired on the existing design. The resulting rule violations are collected, processed to form the textual and graphical contents of the critiques, and returned to the user interface. The interface notifies the designer about the rule violations, and the designer may view the critiques by selecting Critiques... from the action menu.

The review on equipment-layout yields a critique, which the designer views in the Violation window (Figure 52). Figure 53 shows the graphical form of the critique. In this case the critique is actually a simple design suggestion for a hatch to be placed somewhere on the roof. The hatch provides easy access to the roof and is necessary for maintenance of the roof-mounted equipment. The critique illustrates this by drawing a hatch object in the suggestion window in the upper left of the interface window. An arrow points from the hatch to the highlighted outline of the roof, and the textual portion of the critique again is shown in the dialog window.

## Action 2: Placing a Hatch on the Roof Design

To satisfy this latest critique, the designer now selects a hatch to place on the roof (Figure 54). The object selection causes the CMA to use the error prevention strategy. Results are shown in Figure 55. The revised activation pattern of the DTM is shown in Figure 56. The hatch-layout task is now focus and the chimney-layout task has been demoted to active status. The results of the error prevention strategy are very similar to the results for the chimney selection, except that the newly placed chimney also has an appropriate constraint area.

When the designer places the hatch object at the desired location, the error correction strategy generates the critiques shown in Figure 57. All the resulting suggestions involve connecting the hatch with the other roof-mounted equipment.

### Action 3: Laying Out Walkways on the Roof Design

To satisfy the above suggestions, the designer selects the walkway design object from the object palette. The DTM is updated to reflect the new user action and is used in the same fashion as described for the prior two design actions. Walkways are laid out in a rectilinear fashion. After selecting the start point of the walkway, the designer is allowed to extend the walkway along the x- or y-axis (whichever is perpendicular to the last segment of walkway drawn). Figure 58 shows a possible walkway layout for the roof in the example.

### Action 4: Deleting the Fan From the Roof Design

A functional requirement for the building is changed, which eliminates the need for the exhaust fan near the masonry chimneys. Fortunately, its impact on the roof design is minimal; only the exhaust fan object must be removed from the roof plan. The designer removes the object by selecting Delete Object from the action menu and clicking on the exhaust fan. This action results in the roof plan shown in Figure 59.

When an existing object is deleted from the design, the system removes not only the object from the design representation but also all semantic links and other cached information involving the deleted object.

### Action 5: Drainage System Layout

Since the roof in the example is a flat roof, the sloping of the roof to transmit and drain water is solely the responsibility of the roof designer. On a low-slope roof, the heights of the column lines themselves play a significant role in determining the layout of the drainage system. As discussed in Chapter 5, the primary philosophy of roof designers is to place their drains (if using an interior drainage system, as is the case with this example) at points of maximum deflection in the roof field. For example, roof drains should never be placed near load-bearing columns on flat roofs because after settling the roof tends to be higher near the columns. Figure 60 shows a possible drainage system layout that tries to fulfill these requirements. The

drains are halfway between load-bearing members, and all areas of the roof are sloped. The arrows point in the direction of greater deflection.

Figure 61 shows the activation pattern of the DTM after the user has selected a four-slope drainage area (an inverted pyramidal drainage slope). The drainage-area-layout task is now focus, and all the subtasks of equipment-layout are active.

The designer performs a review on the drainage system by selecting the drain-age-system-layout goal from the Goals window and clicking on the Review button. The review yields some critiques that suggest placing two-slope drainage areas (crickets) on the upslope side of the roof-mounted equipment. Without crickets to direct the flow of water around the equipment, water will accumulate (and form pools) on the upslope side of rectangularly-shaped equipment.

## Chapter Summary

This chapter presented how a roof designer might use SEDAR to support the design process in a flexible way. First, SEDAR tries to prevent errors from occurring by graphically marking off-limits areas on the design for a selected design object. Second, SEDAR notifies the designer as soon as an error is detected. This notifica-tion lessens the possibility of extensive redesign if the error were to be discovered later in the design process. Finally, SEDAR allows designers to integrate reviews based on building subsystems (goals in the DTM) seamlessly with the design process. Unlike the current system of design and review, in which reviews may take months, the design review critiquing strategy allows the designer immediate feedback as to whether their design satisfies the basic requirements described by published design specifications.
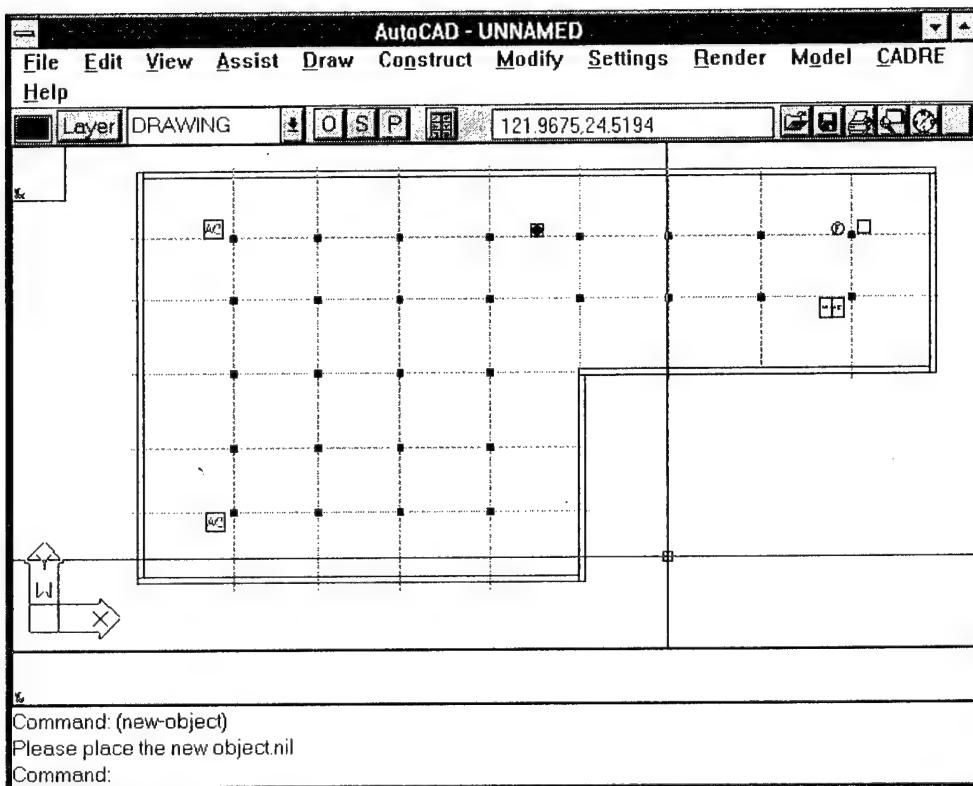
**Figure 40. Initial roof plan.**
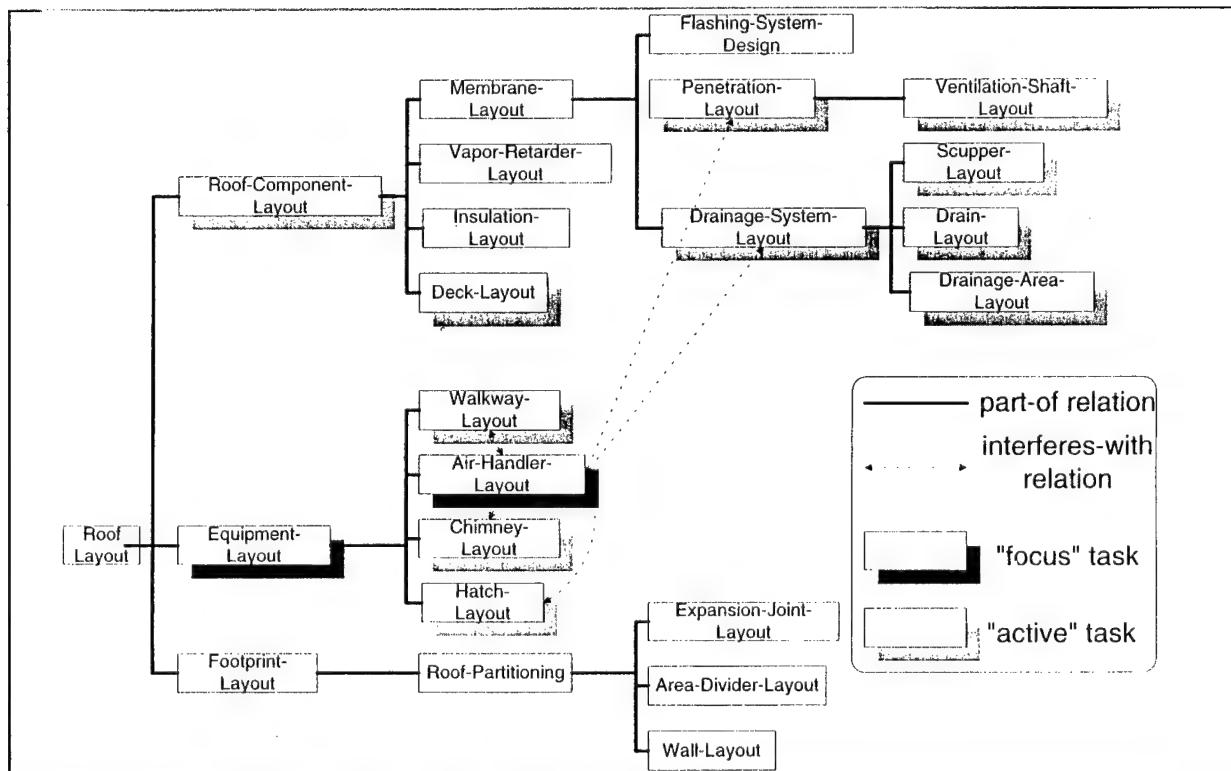


**Figure 41. Initial activation pattern of the Designer's Task Model.**
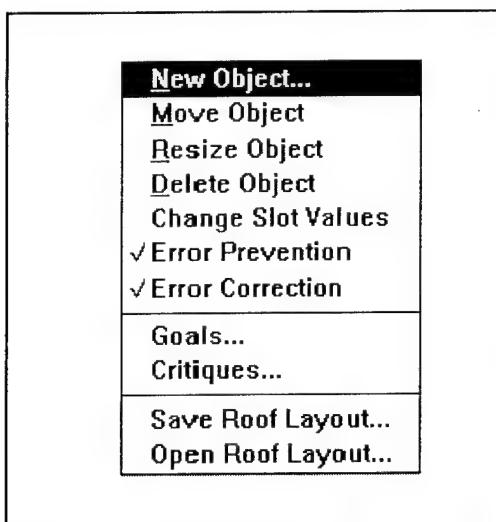
**Figure 42.  The Action Menu.**
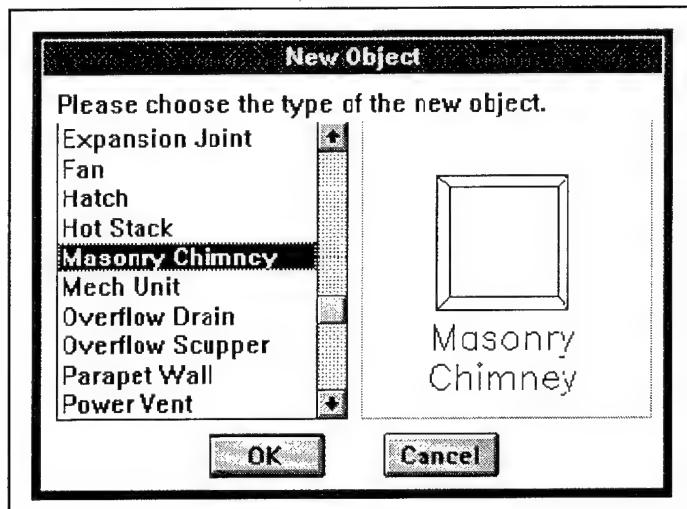


**Figure 43.  Selection of a masonry chimney.**
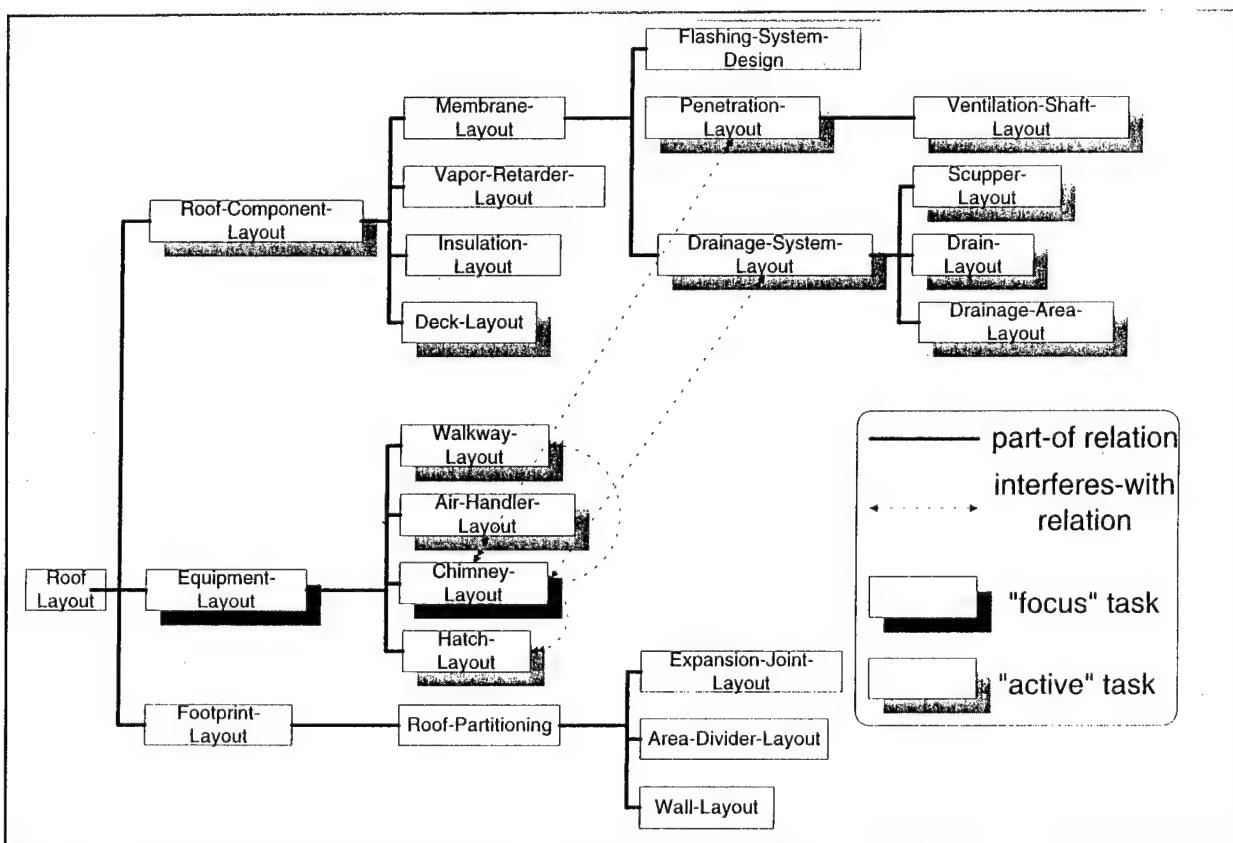


**Figure 44.  Activation pattern of the DTM after masonry chimney selection.**
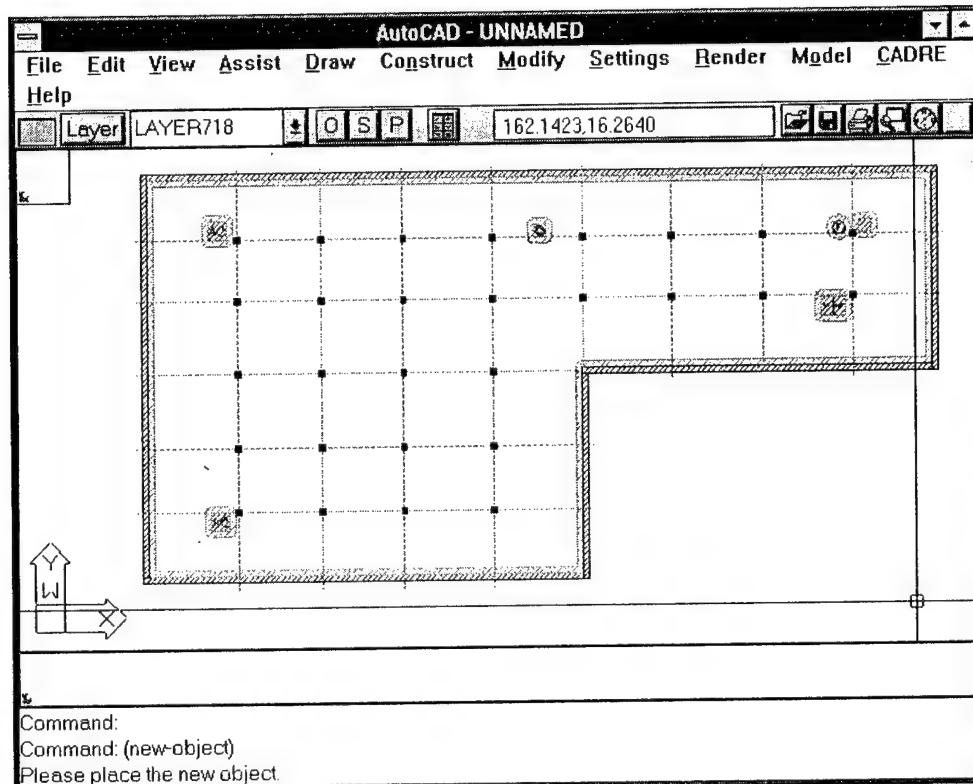
Figure 45.  Results of the error prevention critic.
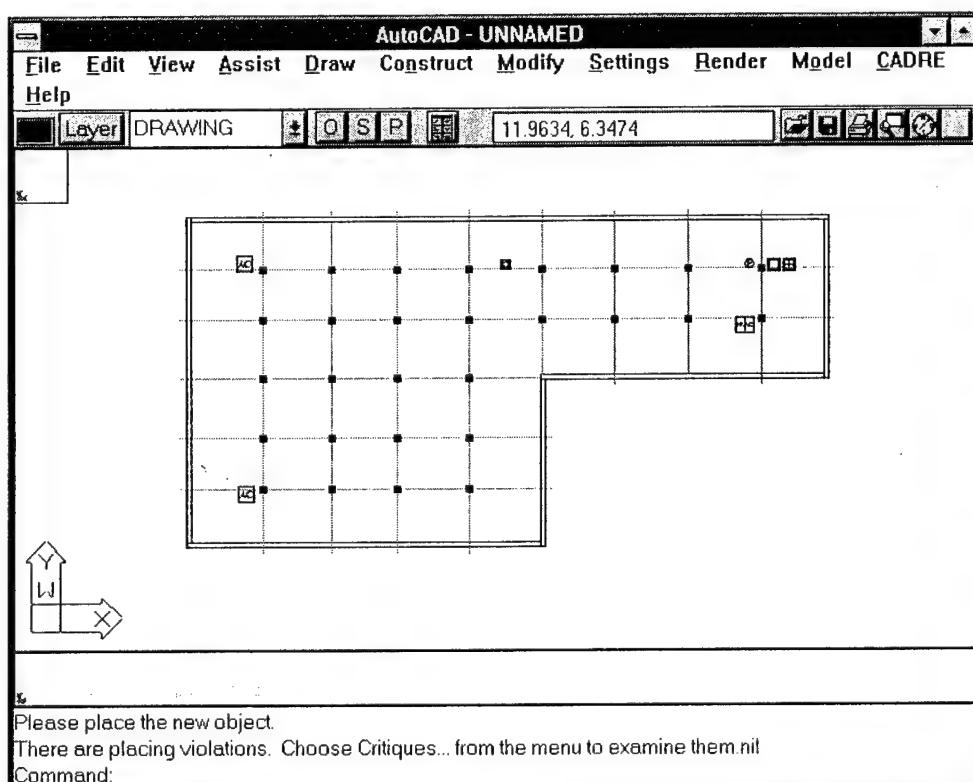


Figure 46.  Results of the error correction critic.
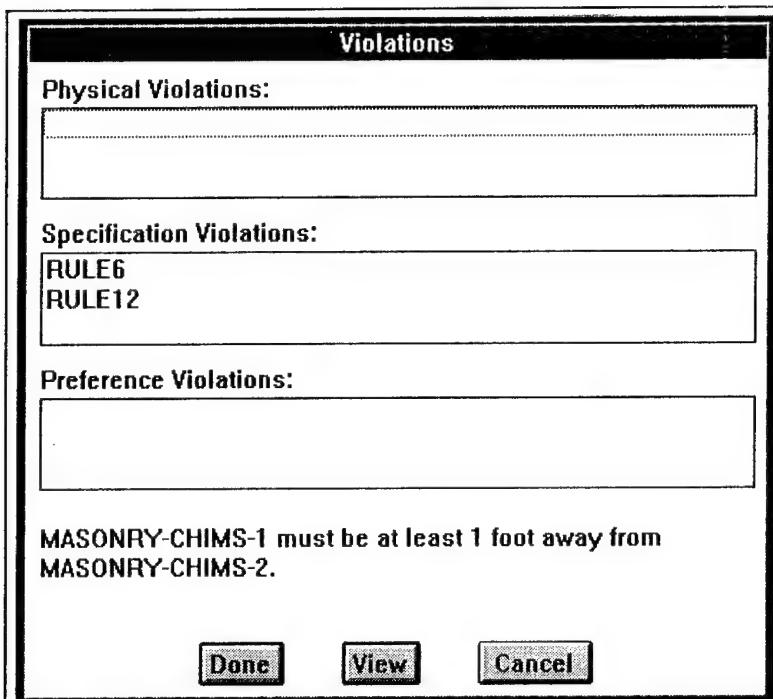
**Figure 47.  The Violations window.**
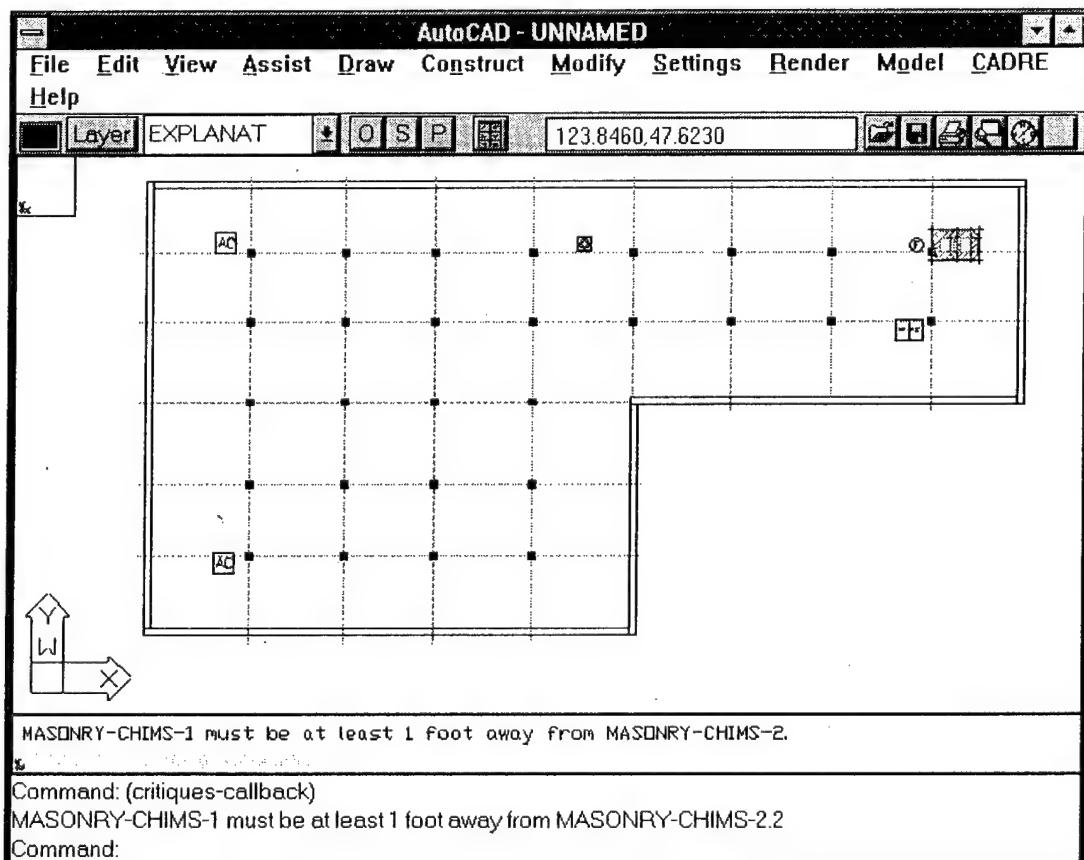


**Figure 48.  The graphical portion of the RULE6 critique.**
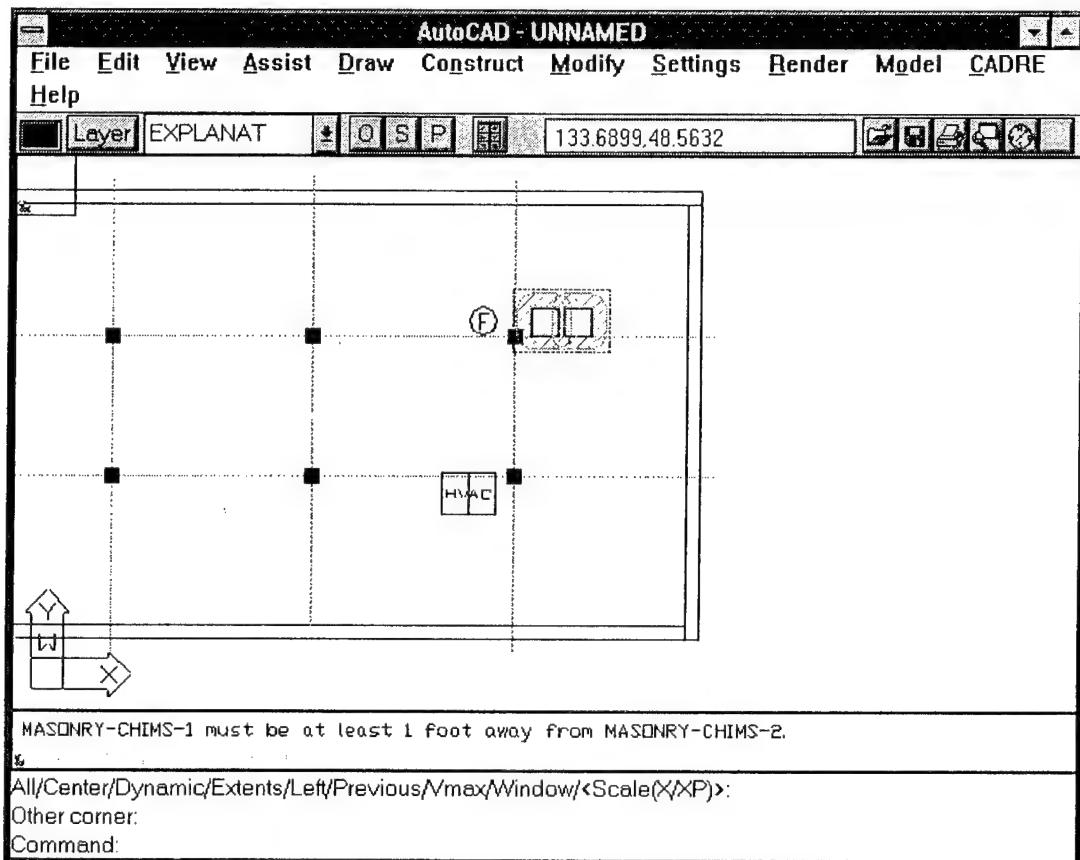
Figure 49.  Enlarged view of the graphical portion of the RULE6 critique.
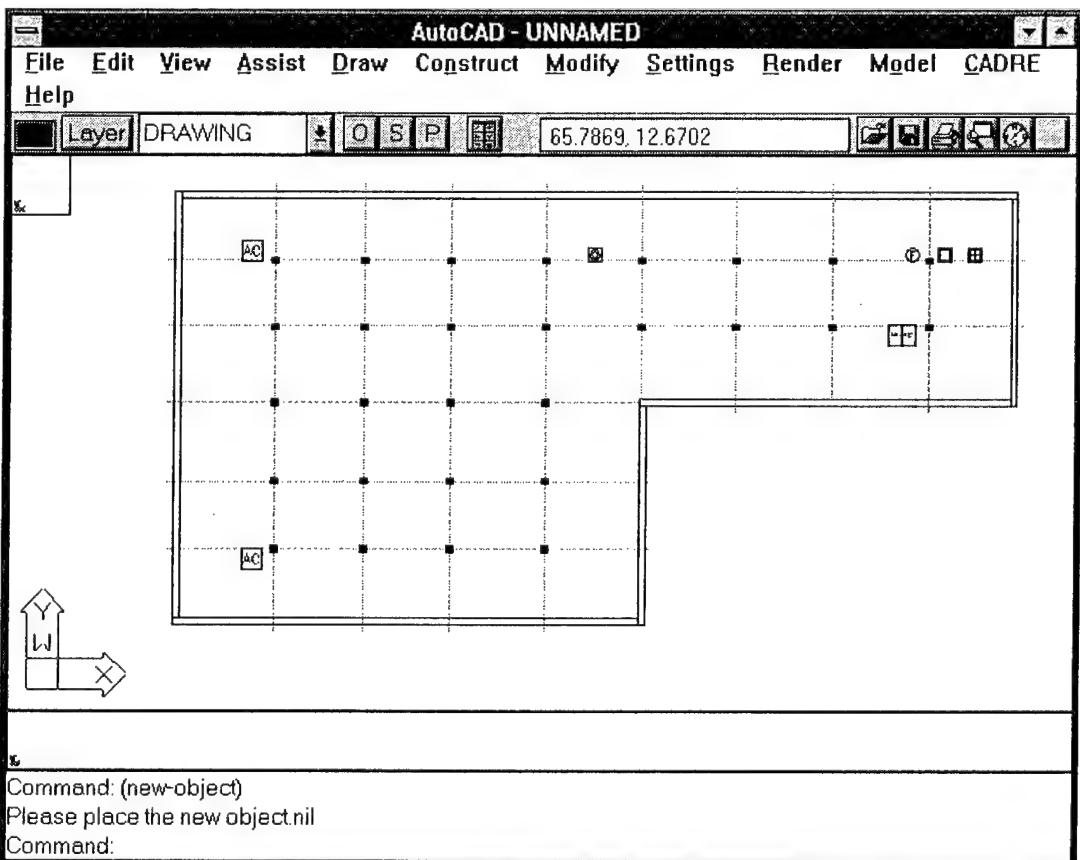


Figure 50.  Designer's revised chimney location.

| Goals | | |
|---|---|---|
| **Activation** | **State** | **Name [hierarchy]** |
| | On | DECK-LAYOUT |
| | On | VAPOR-RETARDER-LAYOUT |
| | On | INSULATION-LAYOUT |
| | On | MEMBRANE-LAYOUT |
| | On | PENETRATION-LAYOUT |
| | On | VENTILATION-SHAFT-LAYOUT |
| | On | HATCH-LAYOUT |
| | On | DRAINAGE-SYSTEM-LAYOUT |
| | On | DRAIN-LAYOUT |
| | On | DRAINAGE-AREA-LAYOUT |
| | On | SUMP-LAYOUT |
| | On | SCUPPER-LAYOUT |
| | On | FLASHING-SYSTEM-DESIGN |
| | On | EDGE-FLASHING-DESIGN |
| | On | EQUIPMENT-FLASHING-DESIGN |
| | On | ROOF-COMPONENT-FLASHING-DESIG |
| | On | **EQUIPMENT-LAYOUT** |
| | On | AIR-HANDLER-LAYOUT |
| | On | WALKWAY-LAYOUT |
| | On | CHIMNEY-LAYOUT |

Done   Review   On   Off   Cancel

Figure 51.  Selecting a goal to review.

| Violations |
|---|
| **Physical Violations:** |
| |
| **Specification Violations:** |
| RULE12 |
| |
| **Preference Violations:** |
| |
| ROOF-FOOTPRINTS-1 must have a hatch to allow access. |

Done   View   Cancel
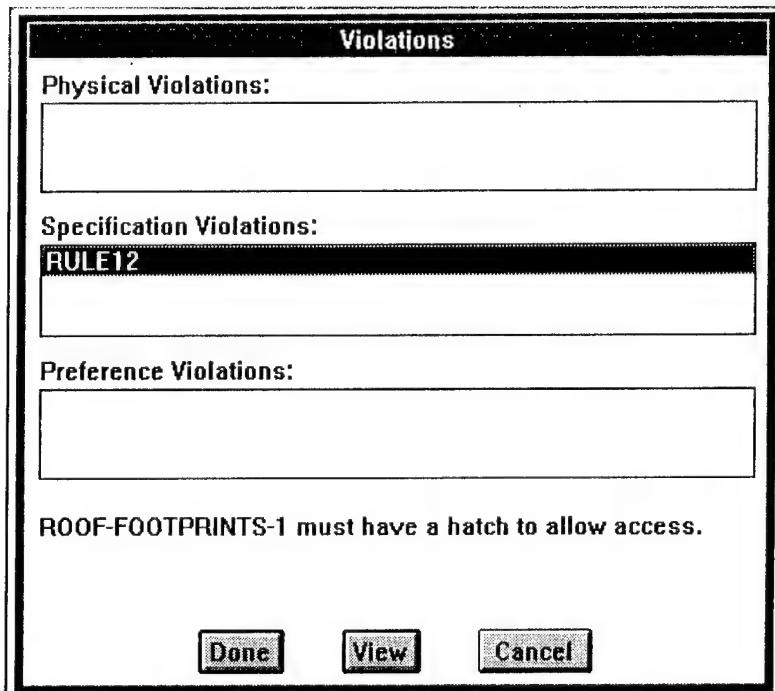
Figure 52.  A Violation resulting from the review.
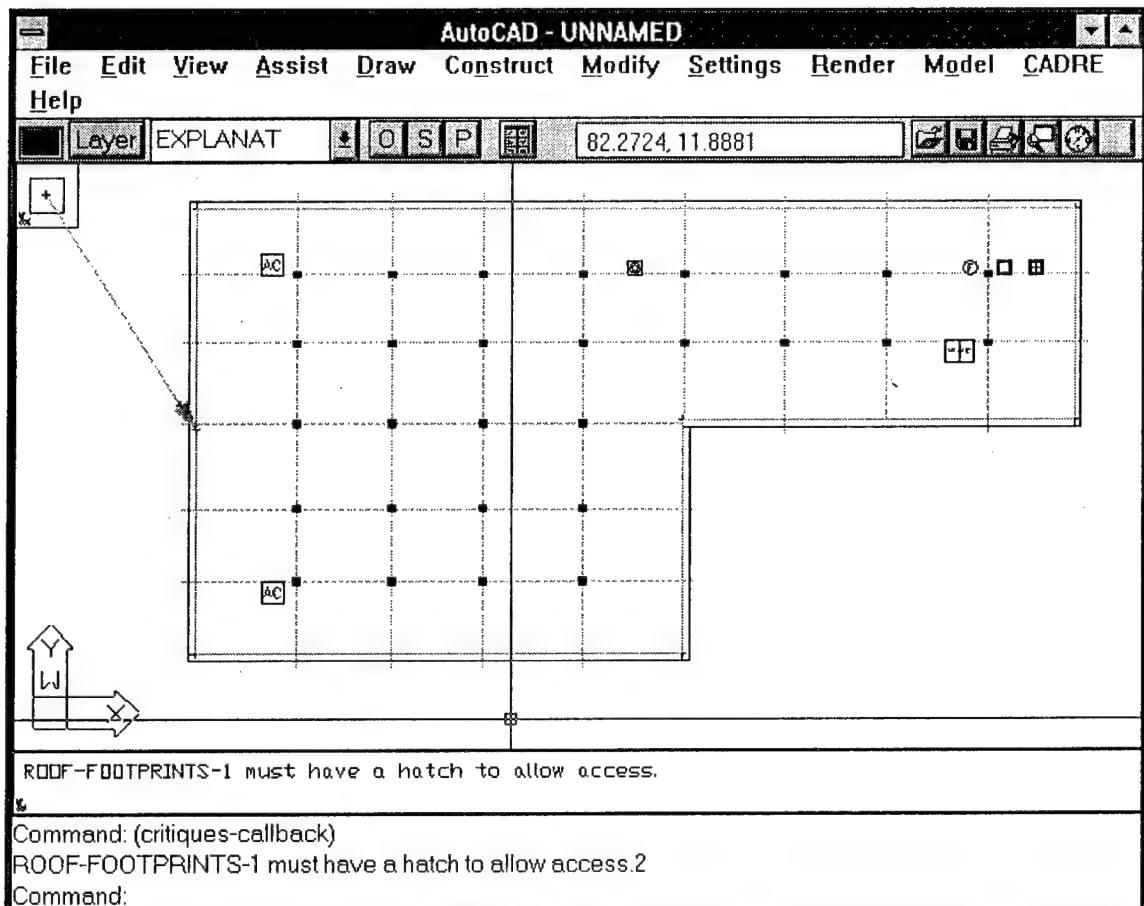
Figure 53.  The graphical portion of the violation.
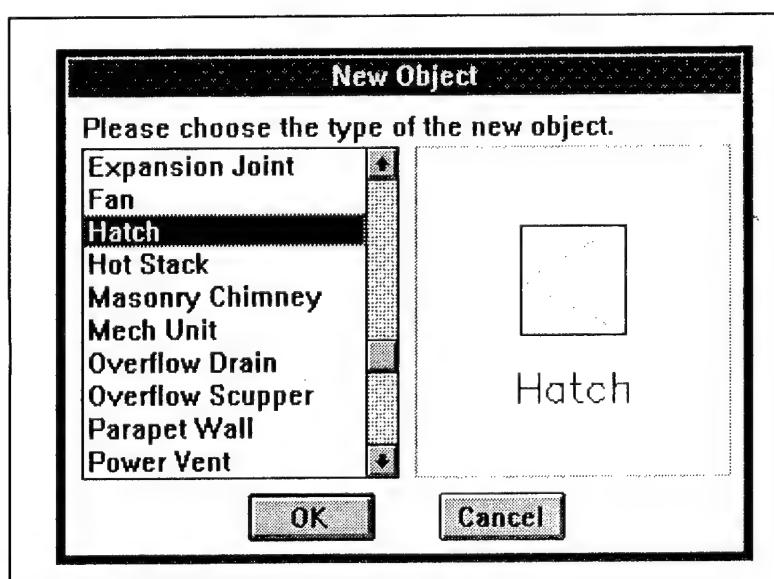


Figure 54.  Selecting a hatch object.
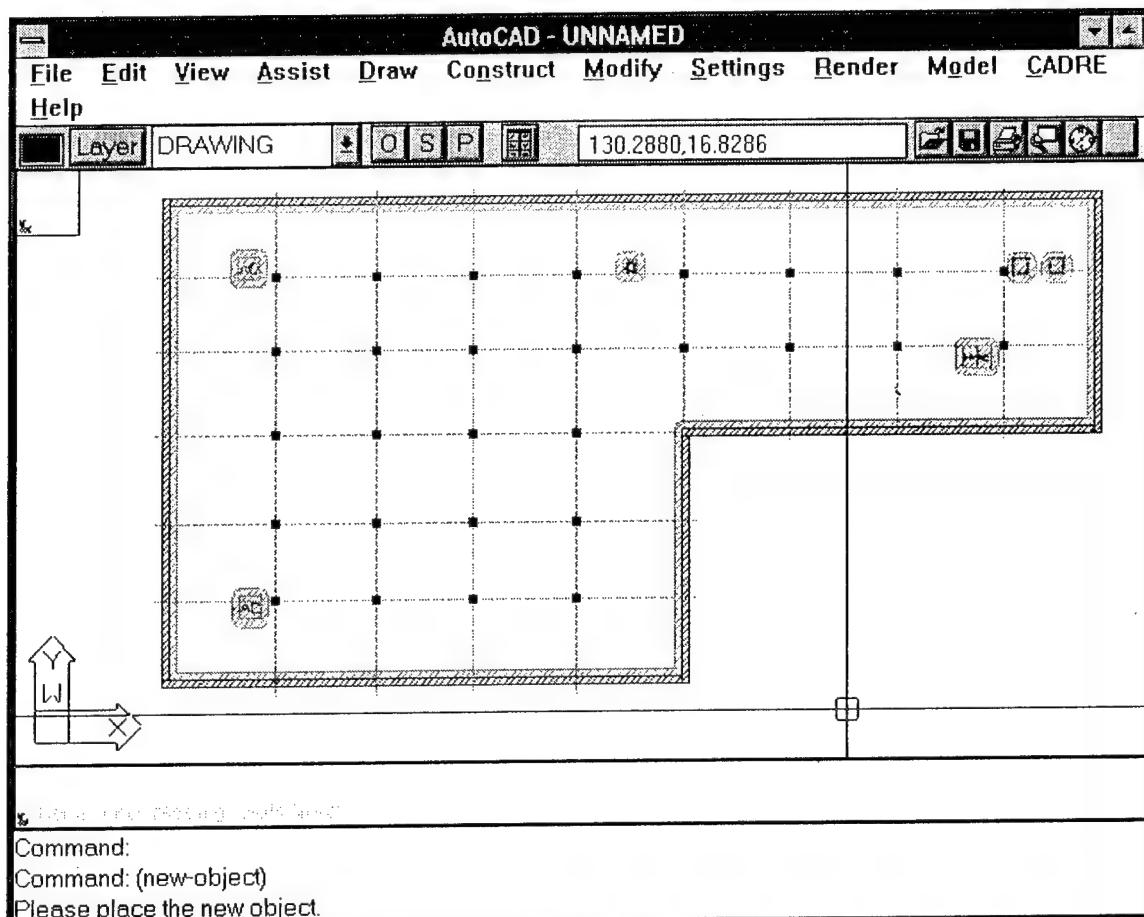
Figure 55.  Results of the error prevention strategy.
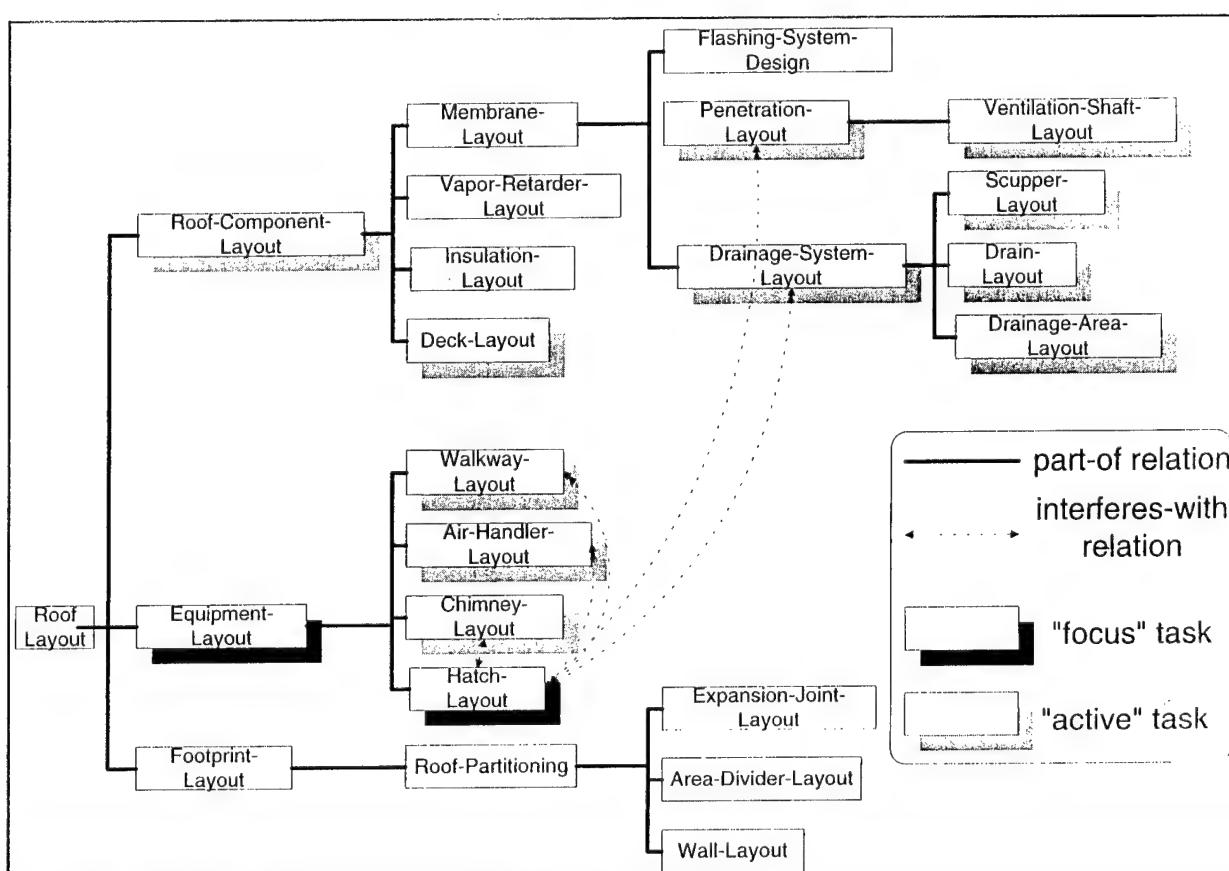


Figure 56.  Activation pattern of the DTM after roof hatch selection.

**Violations**

Physical Violations:

Specification Violations:

RULE21
RULE21
RULE21

Preference Violations:

There should be a walkway from AC-UNITS-CURBED-1 to HATCHES-2.

| Done | View | Cancel |
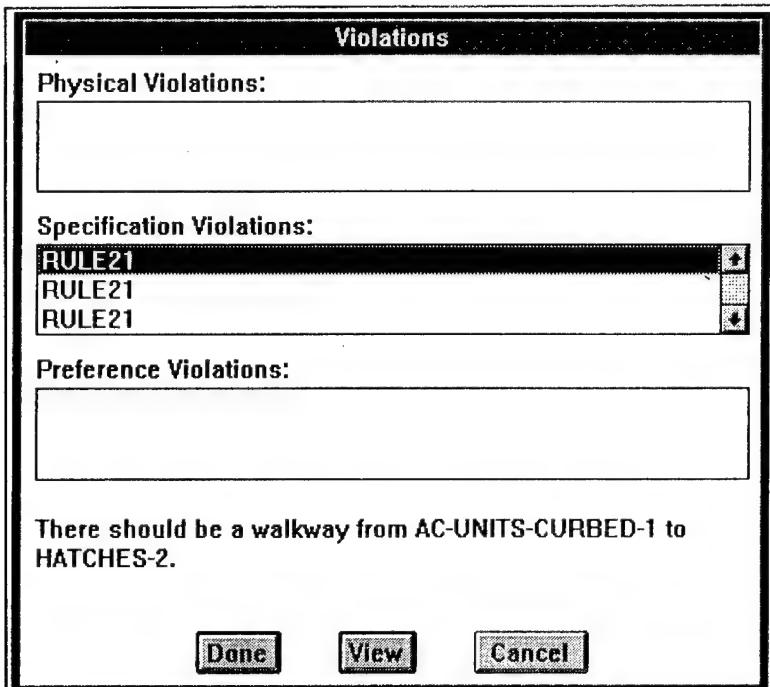
Figure 57.  Results of the error correction strategy.

**AutoCAD - UNNAMED**

File   Edit   View   Assist   Draw   Construct   Modify   Settings   Render   Model   CADRE
Help

Layer   DRAWING          O   S   P          131.1011,32.3044

Command: pan
Displacement: Second point:
Command:
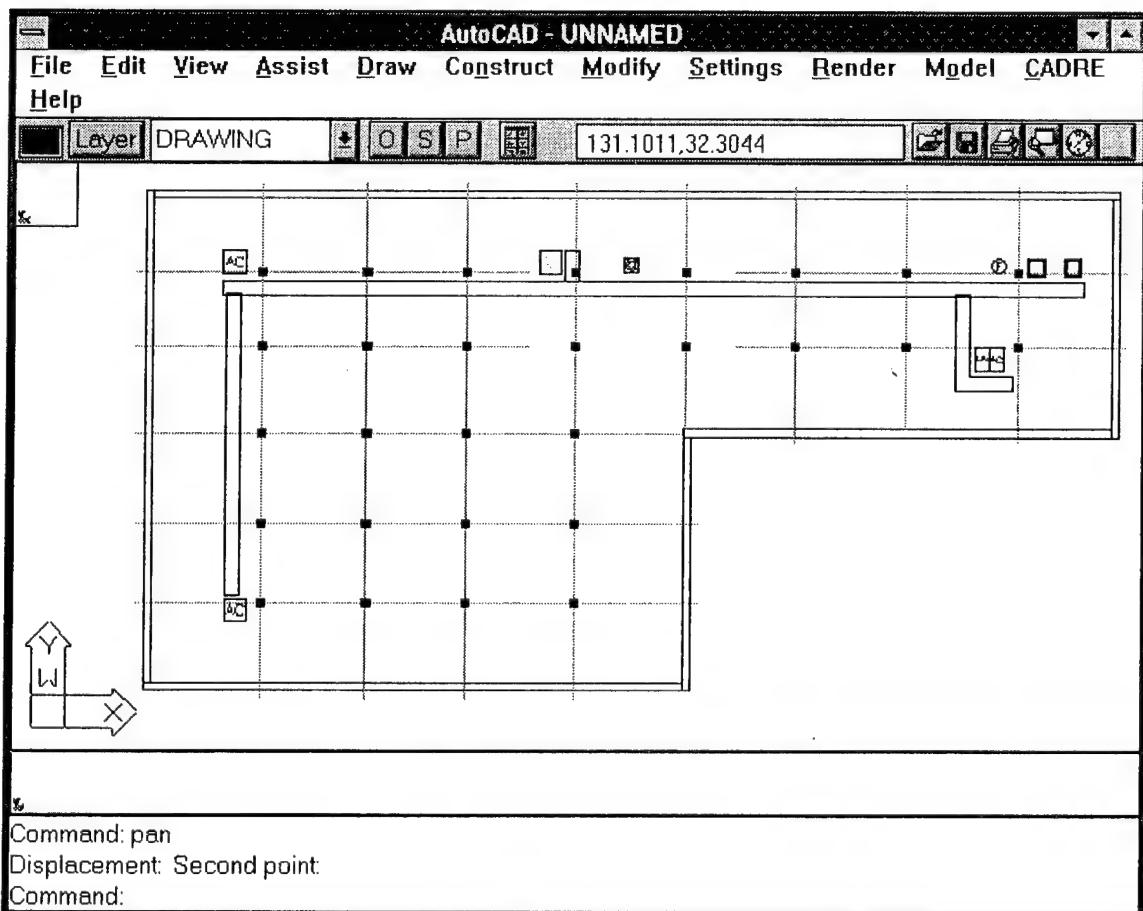
Figure 58.  A possible walkway layout.
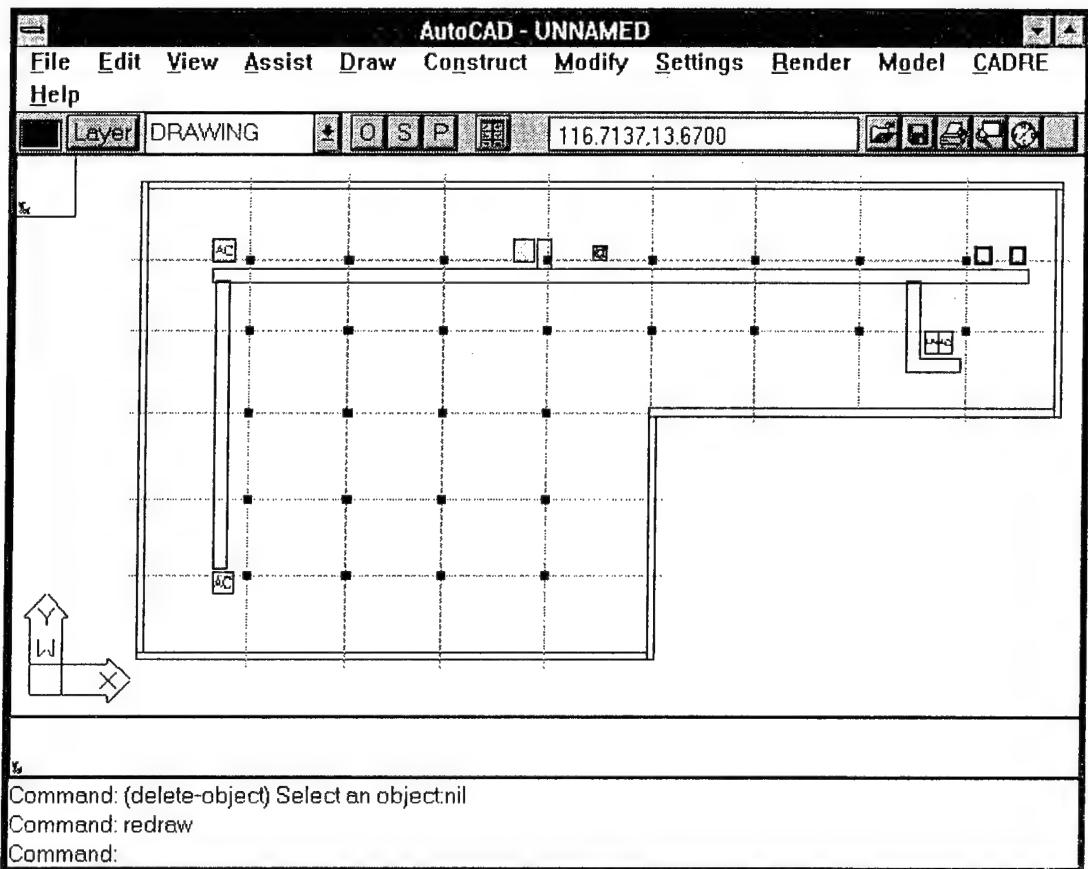
Figure 59.  Revised roof layout after deleting exhaust fan.
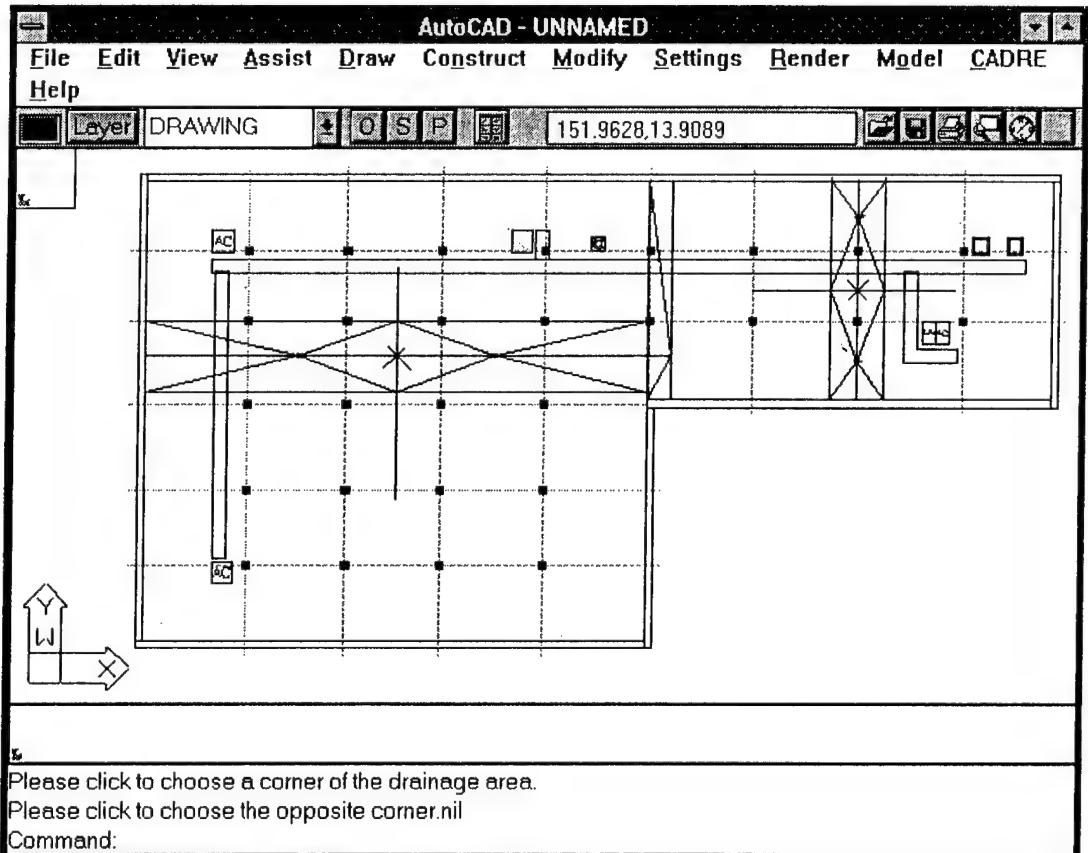


Figure 60.  The drainage system layout.
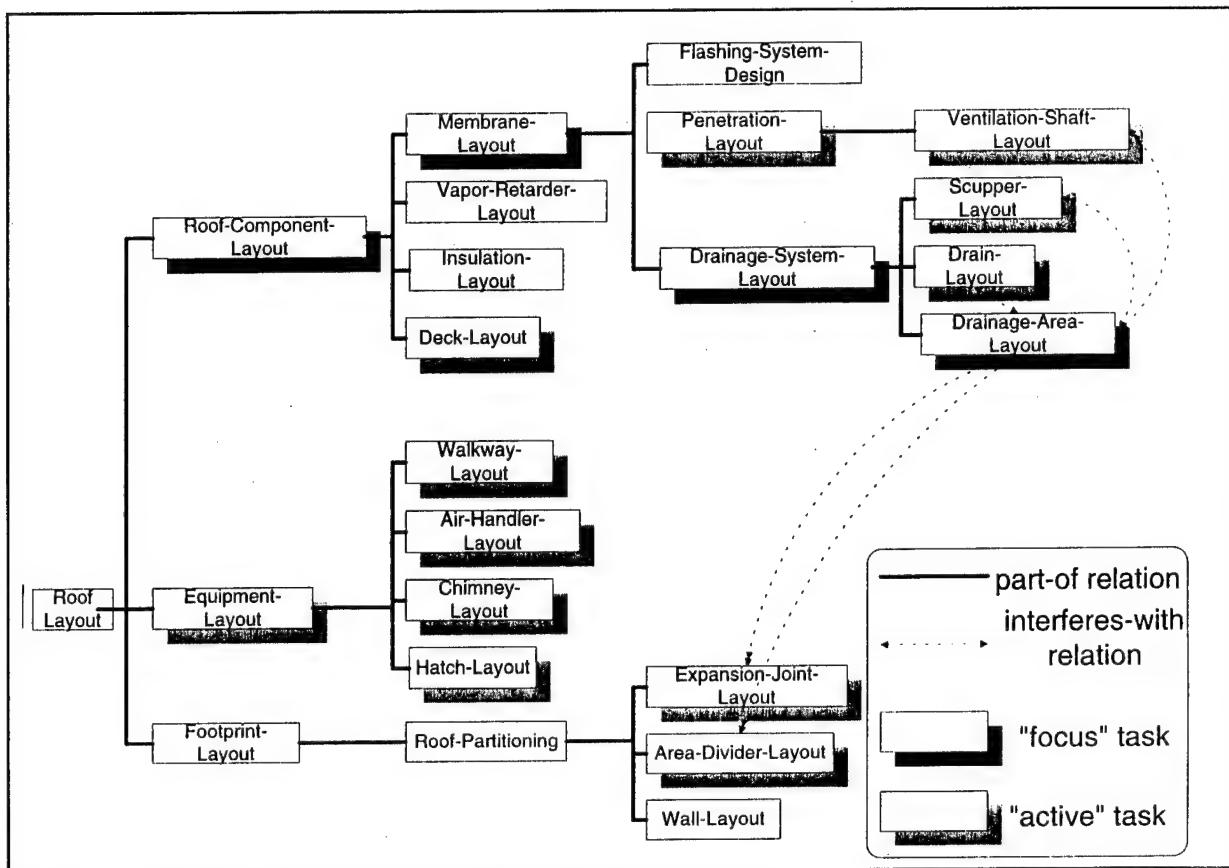
Figure 61. Activation pattern of the DTM after drainage area selection.

# 9  System Evaluation and Testing

After 1 1/2 years of development, a research prototype of SEDAR was available for early evaluation and testing. The need for early evaluation is especially important in interactive systems such as SEDAR, despite the associated difficulties of testing incomplete prototypes. The benefits of prototype evaluation include:

- Discovering which system features assist the target user community and what changes are necessary to make them more effective
- Exposing problem areas in the system design or implementation to be resolved in the next version of the system
- Canvassing of the user community to learn what system capabilities should be developed in the future.

The difficulties associated with prototype evaluation are:

- Deciding when to test the prototype—if the evaluation is conducted too early during development, bugs may distract the focus of the evaluation from the important system features
- Spending valuable resources on evaluation that would otherwise be spent on system development
- Making ad hoc system design decisions due to time pressures that may later create problems.

Thus a system developer performing a prototype evaluation must consider the goals, scope, and content of the evaluation carefully. Realistic goals must be set given available resources and the extent of system implementation. The scope of prototype evaluations is best limited to a small group consisting of members of the targeted user community. This limitation allows the developer to have a deeper, more focused discussion on the evaluation issues.

The first section of this chapter describes the initial evaluation and testing goals for the SEDAR prototype. Experiments to evaluate system usability and the error reduction effectiveness of the prototype are described in the second section. Finally, the results of the two experiments are presented.

## Evaluation and Testing Goals

Expert critiquing systems such as SEDAR may be evaluated along many dimensions, including system usability, error reduction effectiveness, decrease in problem-solving time, range of problems covered, solution feasibility (if the critiquing system has solution generation capabilities), and solution quality.

For the SEDAR evaluation, system usability and error reduction effectiveness were chosen as the primary goals. System usability was chosen because it is a major factor determining whether the target user community (here intermediate- to proficient practitioner-level roof designers) will be willing to use the system in their practices. Error reduction effectiveness was selected because it is the fundamental goal of SEDAR. An early assessment of how well SEDAR performs with respect to these two goals will lead to a better understanding of both the immediate needs of the system and future research directions.

The general issue of system usability was divided into two smaller issues: critiquing strategy usability and interface usability. Critiquing strategy usability refers to how system users perceived usefulness of the critiques provided by the system. This strategy includes questions about whether the system offers the right type and form of critiquing information for the user's level of expertise, whether the critiquing strategies offer the information in an understandable form, and whether the strategies are helping or hindering the user in the context of the user's problem-solving style.

Interface usability involves a number of issues concerning the user interface. The primary issue was the ease of use of the interface and its appropriateness for roof designers. Due to limitations imposed by the underlying development platform (AutoCAD), there was some concern during early stages of the project about how well the general scheme of user and system interaction supported the needs of the user. For example, the set of operations that could be programmed into the system that allow the user to manipulate the roof design was severely restricted due to limitations in the interface capabilities of AutoCAD. Another issue that was a concern during development was the amount of time that the incremental critiquing strategies spent for inferencing and that the user spent waiting for the system to finish its computations.

The second major issue tested was to rate the current system's error reduction effectiveness. Although the system was still a research prototype, with known bugs in the knowledge base, a preliminary test of this issue would show the areas needed for improvement in this area. Another issue that was a concern was the usefulness

of the type of assistance provided by the program to proficient roof designers. As mentioned earlier, the design code specifications reflect the minimum standard of quality for roof design. These specifications say very little about correct but sub-optimal designs. Thus the current system is unable to evaluate and to critique the optimality of these types of designs.

The two issues above are the most relevant to the current stage of system development. Both issues address the basic method of interaction between system and user, which should be validated before further development takes place. Possible future issues to be considered are to decrease in problem-solving time and the range of problems covered by the system. Determining the cost benefits of using SEDAR requires an experiment comparing the time spent in the design/review process between a group using the critiquing system and a group without the critiquing system. Testing in this environment requires the system to be beyond the research prototype stage.

Another dimension for evaluation is the range of problems the system can critique. For example, the current system cannot represent nonrectilinear roof designs. Developing a representation (and geometric reasoning routines) for arbitrary two-dimensional shapes would improve the range of problems that the system can critique, but evaluation of this issue is not appropriate for the research prototype.

## Experiment Descriptions

This section describes the methods used to test system usability and the system's error reduction effectiveness. The system evaluators ranged from novice- to proficient practitioner-level roof designers. First, a profile of the system evaluators is presented that will help explain the results of the experiments and play a major part in the discussion in the next chapter. Following the profile are the descriptions of the experiments. For each experiment the materials presented to the evaluator and the experimental procedure are described. Testing system usability involves a great deal of measuring and noting users' responses to different aspects of the system—the usability experiment involved both protocol analysis and evaluation forms. The error reduction issue was more difficult to test because of the small number of properly qualified roof designers available locally. Testing on this issue involved giving the evaluator two distinct roof design situations of roughly equal difficulty, and having the evaluator perform a roof design on one without the aid of the critiquing strategies and on the other with the critiquing strategies "on." Two values were tabulated for the noncritiqued design—the number of different classes of errors and the total number of errors in the design. Each design code in the knowledge base

was considered a separate class of error. For the critiqued design, the incremental critics (error prevention, error correction) were activated and the user was also allowed to use the design review critic. During the experiment, the number of errors prevented or corrected by SEDAR were recorded.

In the interest of making the evaluations as concise as possible, the two experiments were run partially simultaneously. The designers were asked to make comments about the usability of the system throughout their work on the two roof designs. After the roof designs were completed, or terminated due to time constraints, a short debriefing question and answer session was conducted and the evaluators were asked to fill out the evaluation forms.

### The System Evaluators

The system evaluators were professional architects with experience in roof design and review and were employees of the U.S. Army Corps of Engineers. Some architects in this group had less than 1 year of roof design expertise, and some had a number of years of practice. The wide range of expertise was chosen to determine whether the level and type of assistance provided by the program was appropriate for the targeted practitioner- and proficient-practitioner-level roof designer. Of the six people participating, two had the appropriate level of expertise (practitioner and proficient practioner), two had architectural backgrounds but were not practicing roof designers (intermediate level), and two had little or no roofing experience (novice level). The roof designs of the intermediate- and novice-level evaluators were not used in the error reduction experiment.

### The System Usability Experiment

Three techniques were used to assess the system's usability. First, the evaluators were asked to comment on the system as they worked on the two roof designs. They were also prompted at times to explain their reasons for performing certain actions. Second, after the designs were completed, a short verbal question and answer session was conducted. During this session, they were encouraged to expand on their earlier comments. Finally, they were asked to fill out an evaluation form. The evaluation consisted of roughly 30 specific questions about system usability and their background in roof design. Fill-in-the-blank, short answer, and numerical ratings were used. The evaluation form was divided into five sections:

1.  Background—The evaluators were asked to assess their expertise in roof design and describe their architectural background and experience in using computer-based design programs (i.e., CAD programs).

2.  System Usability: The User Interface—The evaluators were asked to rate and to make comments about the usability of the interface. This section included the ease of use and cuing issues discussed earlier. A series of questions about the clarity and expressiveness of the critique display strategy was also included in this section.

3.  System Usability: The Error Prevention Critic—The evaluators were asked to rate and make comments about the usefulness of the error prevention critic. In particular they were asked to evaluate the usefulness of the information provided by the critic and whether it had helped or was detrimental to their problem-solving process.

4.  System Usability: The Error Correction Critic—This section contained questions very similar to the section on the error prevention critic.

5.  Overall—The evaluators were asked whether they felt the critiquing model fit the goals of the system: supporting the design/review process. They were also asked if they would personally use the system if it were developed further.

### The Error Reduction Experiment

For this experiment, two roof design tasks were given to the evaluator, who worked on the first roof design in the SEDAR environment without any critiquing support and on the second roof design with all of the critiquing strategies activated. Which roof design task was given first to the evaluator was random. The number of errors made by the evaluators on each of the designs, with and without the critiquing strategies activated, were tabulated.

Figure 62 shows the first roof design, which is divided into two architectural zones. The second roof design, which is shown in Figure 63, has three architectural zones. For each of the tasks, the roof designer was asked to complete the design, adding whatever roof-mounted equipment, drainage slopes, etc. necessary.

## Results

The results of the experiments proved to be highly instructive, exposing the strengths and weaknesses of the current implementation. The first part of this section describes the outcome of the system usability experiment, combining results from the evaluation forms, protocol analysis, and the verbal question and answer session. Due to time and resource limitations, the original scope of the error
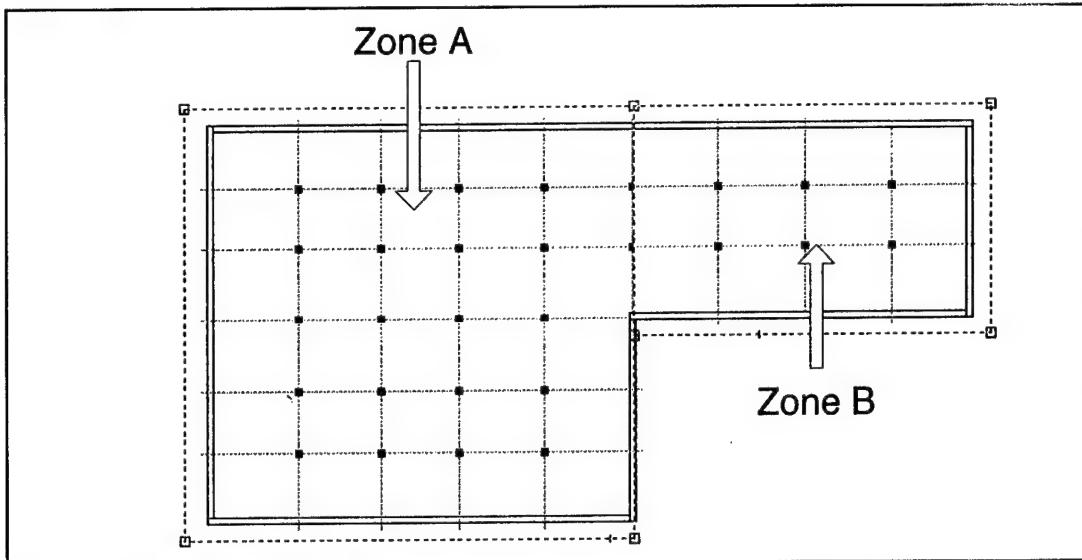
**Figure 62. Roof Design Task 1.**
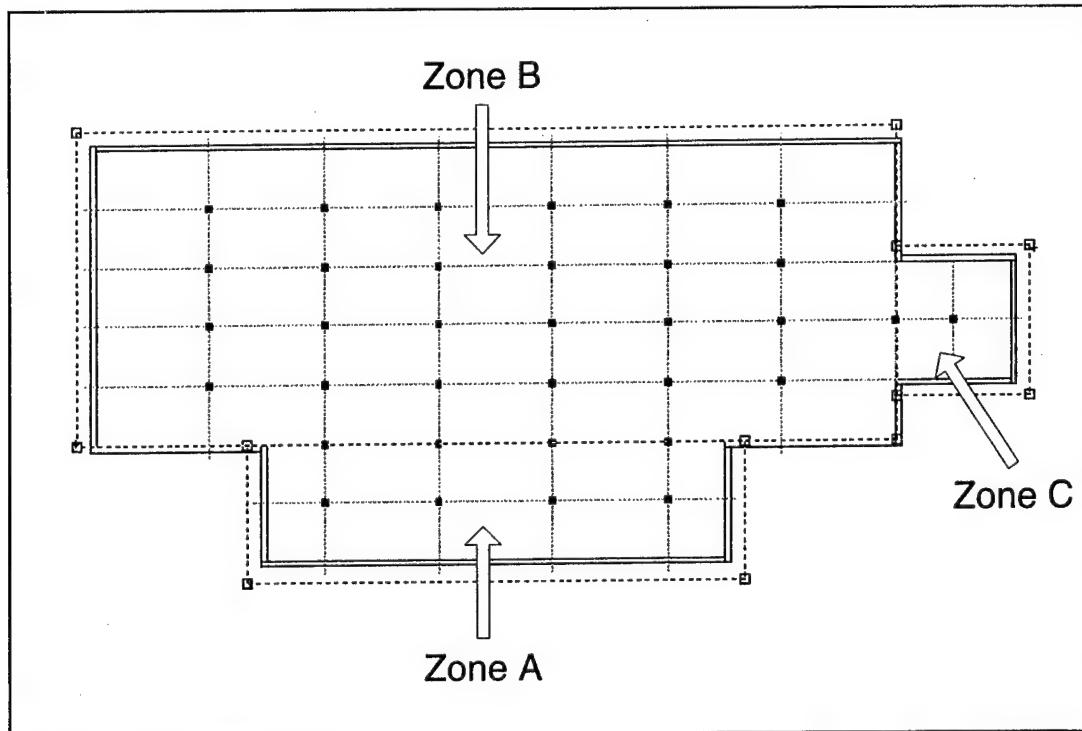


**Figure 63. Roof Design Task 2.**

reduction experiment was limited to two datapoints (the two experienced roof designers).

## The System Usability Experiment

As expected, a great deal of information was available regarding system usability. The roof designers, not surprisingly, had the most to say about how well the system

supported the design/review cycle. The novice- and intermediate-level roof designers provided feedback on the interface's usability and the effectiveness of the incremental critiquing strategies.

*Interface usability.* Figure 64 shows the average values of evaluator responses to evaluation form numerical rating questions pertaining to interface usability. For each of the questions, the average value is circled.

The overall usability ranking of the interface (Question 1) was influenced negatively by two factors. First, when the incremental critiquing strategies were activated, the system slowed down dramatically—after repeated object placements the roof designers became impatient. This issue is intertwined with the issue of critiquing strategy usability, discussed later in this chapter. Second, the designers tended to focus on the drawing window and missed critical textual information in the dialog window. Since the dialog window was the primary source of information about system readiness, users were often unsure whether critiques were available, when the system was ready for them to place an object on the design, and how the objects were to be placed. This and related issues are described in the next chapter. Despite these particular negative comments, system users generally were satisfied with the interface's overall usability.

Question 1: What is your overall assessment of the usability of the current interface?

| Not usable at all | | Neutral | | Very usable |
|---|---|---|---|---|
| 1 | 2 | (3) | 4 | 5 |

Question 2: How difficult/easy was it to learn how to use the interface?

| Difficult | | Neutral | | Easy |
|---|---|---|---|---|
| 1 | 2 | 3 | (4) | 5 |

Question 3: What is your assessment of selecting roof objects from an object palette?

| Extreme dislike | | Neutral | | Liked it a lot |
|---|---|---|---|---|
| 1 | 2 | 3 | (4) | 5 |

Figure 64. Evaluation form results.

The direct iconic manipulation scheme of the interface was intuitive and easy to learn for most system users (Question 2). One of the evaluators had never used a CAD system before, and thus had some initial difficulties with the interface. Although he was never as fluent with the system as the other evaluators, at the end of the evaluation session he seemed comfortable with the interface.

Finally, users liked the design object palette. One central concern regarding the interface was whether users would find unacceptable the limited set of roof components available in the object palette. This was not the case, and many stated that having standardized design objects would reduce their drawing effort.

The second set of questions shown in Figure 65 pertain to how SEDAR displays its critiques. While users found the graphical portions of the critiques clear and easy to understand (Question 5), they had a more difficult time understanding the textual portions (Question 6). This was because (1) the terminology used in the textual portions was too specific to the representations in SEDAR (i.e., a cricket was described as a two-slope-drainage-area in reference to its generic object type in

Question 4: How well was the system able to communicate its critiques to you overall?

Not able to                                                          Able to
communicate clearly                                      communiate them clearly

    1              2              ③              4              5

Question 5: How easy was it to understand the graphical parts of the critiques?

Difficult                                                          Easy to
to understand                                                   understand

    1              2              3              ④              5

Question 6: How easy was it to understand the textual parts of the critiques?

Difficult                                                          Easy to
to understand                                                   understand

    1              2              ③              4              5

Figure 65. Evaluation forms results, continued.

SEDAR) and (2) the critiques referred to particular design objects (like two-slope-drainage-area539 or roof-drain56), and users were not able to relate the critique to the objects on the drawing.

***Critiquing Strategy Usability***. The set of questions in Figure 66 pertain to the incremental error prevention and error correction strategies. In general, designers found the incremental strategies helpful (Questions 7 and 9). Novice- and intermediate-level roof designers found the incremental strategies very useful. Experienced roof designers had several complaints about the incremental strategies, which led to a rethinking of how such strategies should be implemented for this category of users. In addition to the slowness of the system operation when running the incremental critics, the evaluators also pointed out problems with the granularity of the strategies with respect to their design actions. To clarify, SEDAR's critiquing strategies are based about the design-object activity level of user actions—critiques are

Question 7:  How useful was the information provided by the Error Prevention Critic?

| Not useful | | Somewhat useful | | Very useful |
|---|---|---|---|---|
| 1 | 2 | 3 | (4) | 5 |

Question 8:  Is the Error Prevention Critic helpful or detrimental to you?

| Detrimental | | Neutral | | Helpful |
|---|---|---|---|---|
| 1 | 2 | 3 | (4) | 5 |

Question 9:  How useful was the information provided by the Error Correction Critic?

| Not useful | | Somewhat useful | | Very useful |
|---|---|---|---|---|
| 1 | 2 | 3 | (4) | 5 |

Question 10:  Is the Error Correction Critic helpful or detrimental to you?

| Detrimental | | Neutral | | Helpful |
|---|---|---|---|---|
| 1 | 2 | 3 | (4) | 5 |

**Figure 66.  More evaluation form results.**

run every time a new object is placed on the drawing. With this philosophy, the system tends to focus its critiques on individual objects rather than systems of objects, which causes it to provide critiques at incorrect times. An example of this is the hatch requirement design code. As soon as a designer begins to place mechanical equipment in the roof field, SEDAR discovers this "violation" and reports it to the user. However, the evaluators asserted that they fully planned to put a hatch on the roof eventually, and the repeated "violations" were inappropriate at that stage of the roof design. Thus the timing of certain types of critiques (e.g., design suggestions generated by object-existence design codes) need to be improved.

Despite these criticisms, both inexperienced and experienced users felt that the incremental strategies did help them to perform their task (Questions 8 and 10). This point is further evidenced by the results of the error reduction experiment reported in the next section.

### The Error Reduction Experiment

Preliminary results from the error reduction experiment show that SEDAR reduces the number of errors made by roof designers. The total number of errors and the different classes of errors made by each designer were tabulated and are reported in this section. A class of error represents all errors resulting from the application of a particular design code to the drawing. For instance, if the situation in Figure 67 occurs, there are three total errors (each pair of air-conditioning units are too close together) but only a single class of error, since each error was the result of the application of the same design code.
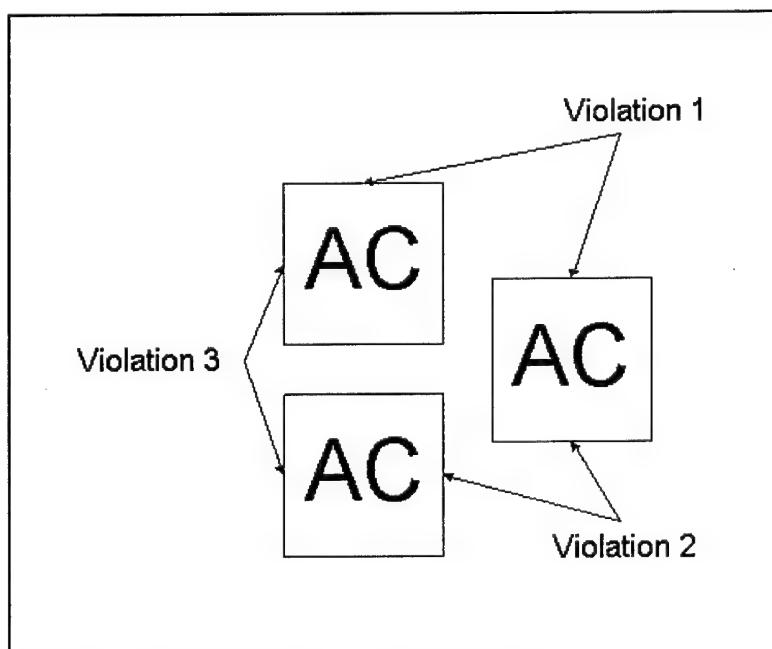


Figure 67. Three total errors; one class of error.

One roof designer made four classes of errors and 15 total errors during the non-critiqued design task. During the critiqued design task, the same designer made five classes of errors and 15 total errors. The system cued the roof designer properly on four of the five classes of errors and 14 out of the 15 total errors.

In another experiment, SEDAR was able to cue the roof designer on four out of five classes of errors for a total reduction of seven out of eight errors. The system failed to cue the designer on an optimization issue regarding the closeness of an air-handling unit to a roof drain; although the placement of the unit satisfied the relevant design code, the unit (or the roof drain) was placed in a suboptimal location. This issue is discussed in the following chapter.

In general, the results show that roof designers will benefit from the use of SEDAR. Specific problems with the knowledge base involving completeness and correctness were exposed and will be dealt with in future versions of the system.

# 10  Discussion

Many issues were raised during the evaluation and testing process of the SEDAR system prototype. For the sake of conciseness, only the most interesting are discussed here. Four issues that arose during the evaluation involved system usability. The first issue directly involves some of the questions about the system's usability: the value of the error prevention, error correction, and design review critiquing strategies implemented for the prototype. The second issue, that of a strong ordering of tasks in the roof design domain, arose during the question and answer process and is discussed in the second section of this chapter. The third issue involves the notion of optimizing rather than satisficing designs. Specific facets of the user interface design elicited strong comments from the evaluators, and these are described in the fourth section of this chapter. Finally, many of the evaluators expressed a concern regarding the adaptability of the system—whether it had the capability to either add new rules by itself (machine learning) or to allow users to enter new rules (knowledge acquisition).

## On the Usability of Critiquing Strategies

The three practitioner/proficient practitioner level evaluators had strong reactions to the various critiquing strategies. The general consensus among these evaluators was that the incremental error prevention and error detection strategies as implemented would be of little interest to them. However, they were not adverse to the idea of incremental critiquing. The problem was with how the iterative critiquing cycle worked. Specifically, creating a set of constraint areas (as the error prevention critiquing strategy does) each time a design object was selected was annoying. Similarly, they were not pleased with the way the error correction strategy critiqued the design when a single new object was added to the design. Essentially what is needed is a less "continuous" style of critiquing than the current implementation. The system should form its critiques based on collections of objects rather than each object in isolation. For example, one evaluator did not like the "roof requires a hatch" critique he encountered repeatedly while placing air-handling units on the roof design. He argued that he fully intended to put a hatch on the design, so there was no need for a constant stream of critiques to remind him of that fact. Therefore, the timing of critique generation for the incremental critiquing strategies should be

improved. Specifically, research is needed to determine when the designer is finished creating certain subsystems so that the critiquing strategies may be applied at the correct time.

In contrast to the somewhat negative response regarding the incremental critiquing strategies, the experienced roof designers liked the design review critic. One possible reason for this is greater control over when the critic is activated. The comprehensiveness of the design review also appealed to them.

The two users at the intermediate level of expertise found the error prevention and error detection strategies more useful than the design review critic. There were two reasons for the disparity. First, the error prevention and error detection critics tended to influence their problem-solving behavior. Looking at the examples in Chapter 8, the contents of the critiques implicitly suggest the next step to the designer. For intermediate-level users who are not yet confident of their problem-solving abilities in the domain, the problem-solving structure defined by the series of critiques is reassuring. Second, the intermediate-level users seldom activated the design review critic, if at all. Their focus was on the design itself and the critiques generated by their actions.

The different critiquing strategies offered by SEDAR appeal to different levels of users. While practitioner and proficient practitioner level designers view SEDAR more as a tool that may be invoked occasionally to help them near the conclusion of their problem-solving, intermediate-level users favor the incremental critiquing ability of SEDAR to guide them through the roof design process.

## Strong Orderings in the Roof Design Problem-Solving Structure

The use of SEDAR's DTM allows it to follow the designer's problem-solving process as flexibly as possible. However, all of the experienced roof designers participating in the evaluation have noted that some tasks are invariably accomplished before others. The temporal orderings stem from how the roof design task is addressed in the overall building design task. Certain facts about the roof are established early in the building design task. These facts include what type of roof (flat or low-slope) is to be designed, the type of membrane (i.e., EPDM, Modified Bitumen, or PVC) is to be applied, the type of deck, the type of drainage system (interior or exterior), and the types of roof-mounted equipment. These facts are established early because they influence the design activity on other parts of the building. For example, the existence of roof-mounted air-handling units means that the mechanical engineer must provide ductwork to and from the unit. The type of roof and deck influence the

design of the roof framing plan. The remaining tasks for designing the roof are usually begun when most of the building design has been completed. The act of deciding these facts about the roof represent the conceptual stage of the roof design.

In summary, while SEDAR's problem-solving support flexibility is necessary for many of the roof design tasks, the structuring of the tasks described above lends itself to additional assistance opportunities. For example, SEDAR could provide assistance at the conceptual stage of roof design by providing advice on how to answer the above questions, and by actively inquiring about them if the roof design continues with incomplete information.

## Optimization Versus Satisficing Design Advice

The design specifications encoded in SEDAR represent minimal qualifications (satisficing conditions) for a roof design. Experienced roof designers attempt to optimize the layout of systems of objects according to a set of metrics. An example of this is hatch placement optimization. The design codes for hatch placement are relatively simple, the access pathway through the roof framing plan to the hatch should be unobstructed. The hatch itself should be not be obstructed by other roof elements. Observation of a human design expert demonstrated additional concerns: the hatch should be placed in a location that minimizes the walkway distances to the other roof-mounted equipment, and the hatch should not be placed in an area of heavy customer traffic.

Ways of approaching this problem are many from the viewpoint of creating critiquing systems to support the designer's reasoning process. One approach is to do nothing to resolve the issue. Another approach is to acknowledge the issue and provide information (not necessarily critiques) about the decision to be made. For example, a checklist of design issues could be shown to the designer. Yet another approach is to have the system critique the designer's solution according to these new functional requirements for the hatch and to actively elicit the answers to the unknown requirements. For this last approach, great care must be taken to create a complete set of functional requirements for each design task.

## User Interface Issues

All of the evaluators had comments to make on the user interface. This section discusses the two most important lessons learned. Designers of all experience levels tended to focus solely on the Design Window, which contains the graphical

representation of the design, throughout the entire session. Very little attention was paid to the text-based Dialog Window. The problem with the current implementation was that nearly all communication from the user interface (except for the graphical critique displays) were in textual form and displayed in the dialog window, where they were ignored by the user. An example of this was the notification that the system was ready to proceed after an object was selected and its constraint areas shown on the drawing. In many cases the user waited patiently until it was pointed out that the system was ready to proceed. Similar situations arose for notifications of available critiques, directions for layout of special objects (e.g., walkways), and textual critique explanations. The suggested remedy was to move the information to the drawing window and make it more graphical in nature, such as placing a large READY button on the drawing window to notify the user when the system is ready for an object placement.

The second lesson is that even incidental wording or terminology used in the system must be carefully controlled. An example of this terminology is the use of the word "violations." The message "There are placing violations found" tended to place the user in an antagonistic frame of mind. The problem is exacerbated when object-existence rule violations are found that specify missing objects on the design; "violations" may be better termed as "suggestions." Thus an interface must be examined carefully to ensure that every communication conveys exactly the intended tone of the critique.

## Knowledge Acquisition and Machine Learning

One of the primary criticisms of expert systems (and hence expert critiquing systems) is that they generally have no provisions for modification of their rule base. While much research has been done in the field of expert systems on knowledge acquisition and machine learning, little research has been done in the field of expert critiquing systems. JANUS [Fischer 1991] is one of the few expert critiquing systems that provide an interface that allows system users to augment the knowledge base. Currently SEDAR has no provisions for knowledge acquisition or machine learning techniques. However, this issue will be addressed in future work on the system.

## Chapter Summary

The results of the evaluation and testing phase on the SEDAR research prototype provide strong directions for future research. While the experienced roof designers

described some problems with the current implementation of the error prevention and error correction critiquing strategies, they provided excellent directions for future research involving the judicious timing and placement of incremental critiques. In contrast, intermediate-level users found the existing incremental critiquing strategies to be more useful than the design review strategy. Interviews with experienced roof designers also established specific optimization considerations, and may lead to a principled design optimization suggestion strategy. An unexpected but welcome result was increased insight into the structure of the roof design task, which may lead to critiquing strategies appropriate for the conceptual design stage. Two important issues regarding user interface design were established. Finally, although knowledge acquisition/machine learning is not addressed in the current system, they are concerns that will be dealt with later.

# 11    Summary and Conclusions

The development of SEDAR is in itself a process of iterative design, subject to the
same fundamental design and review process that it seeks to model and improve.
This chapter summarizes the contributions of this work, the progress to date, the
lessons learned from the evaluation of the system, and directions for future work.

This work demonstrates a technique of using a hierarchically-decomposed, task-
based model of an experienced designer, the DTM, for flexible control of the opera-
tion of an expert critiquing system.  Control is flexible in that the system user
retains full control of the problem-solving process, and is not forced along predefined
solution paths.  Thus SEDAR is able to follow the individual designer's problem-
solving preferences.  At the same time, the DTM allows the system to present the
most focused, appropriate critiques at each step in the user's design process.  Few
user models exist in implemented expert critiquing systems even today, despite their
acknowledged usefulness in these types of systems.  Two systems that do have user
models, LISP-CRITIC and HYDRA, are presented and contrasted to the DTM of
SEDAR.  The primary difference is that the DTM is a process-based representation
of experienced roof designers' task structures, while the other user models focus on
representing the individuality of users' preferences.  Thus SEDAR's use of the DTM
allows finer-grained control over the content and timing of critiques throughout the
design process than other critiquing systems, which is essential for systems that
perform incremental critiquing.

SEDAR may be situated as an architectural design agent in a concurrent engineer-
ing environment, automating the application of constructibility review knowledge
during the design process to roof designs.  This addresses some of the problems of
the current design/review process, where review knowledge is applied infrequently
and often incompletely throughout the design stage of an artifact's lifecycle.  The
integration of design and review provided by SEDAR hopefully will reduce the
length of the design stage, promote more frequent and more complete design
reviews, and reduce the number of errors surviving past the design phase.

The evaluation conducted on the research prototype (Chapter 9) illustrated the
strengths and weaknesses of SEDAR.  The intended user community approved the
design review strategy but did not find the current implementation of the error pre-

vention and error correction critiquing strategies usable. Among the most common factors cited were the length of time for the incremental critiquing strategies to run and the granularity of critique generation and notification. The system evaluators often proposed viable solutions to many of the problems they described, so future work will involve developing incremental critics that better satisfy the needs of roof designers. The error reduction effectiveness study showed that the system can help roof designers prevent and discover errors that are related to design codes in the knowledge base. Another enhancement to the system that was suggested frequently was to have some form of design suggestion in addition to the critiquing strategies. Where the critiquing strategies tell where not to place particular types of objects on the existing design, a design suggestion facility would show the optimal (or near-optimal) location for the object. Finally, another critical issue that was mentioned frequently was knowledge acquisition and machine learning. The ability to add new design objects, design codes, and designer preferences are essential for the use of SEDAR in design offices.

# Bibliography

[Baykan 1992] C. Baykan and M. Fox. "Wright: a constraint-based spatial layout system." In *Artificial Intelligence in Engineering Design*, volume 1, chapter 11. Academic Press, Inc., New York, 1992.

[Brown 1986] D. Brown and B. Chandrasekaren. "Knowledge and control for a mechanical design expert system." *IEEE Computer*, pages 92-100, July 1986.

[Burton 1982] R. Burton and J. Brown. "An investigation of computer coaching for informal learning activities." In *Intelligent Tutoring Systems*, pages 79-98. Academic Press, Inc., New York, 1982.

[Cacciabue 1992] P. Cacciabue et al. "A cognitive model in a blackboard architecture: synergism of AI and psychology." *Reliability Engineering and System Safety*, 36:187-197, 1992.

[Canas 1994] J. Canas et al. "Mental models and computer programming." *International Journal of Human-Computer Studies*, 40:795-811, March 1994.

[Case 1994] M. Case. "The Discourse Model for Collaborative Engineering Design: A Distributed and Asynchronous Approach." PhD thesis, University of Illinois at Urbana-Champaign, 1994.

[Clarke 1991] J. Clarke and D. Randall. "An intelligent front-end for computer-aided building design." *Artificial Intelligence in Engineering*, 6(1):36-45, 1991.

[De La Garza 1992] J. De La Garza and G. Oralkan. "An object space framework for design/construction integration." *Building and Environment*, 27(2):243-255, April 1992.

[Dixon, 1984] J. Dixon and M. Simmons. "An architecture for application of artificial intelligence in design." In *Proceedings of the 21st IEEE Design Automation Conference*, pages 634-640, 1984.

[Dixon 1987] J. Dixon et al. "Dominic I: Progress toward domain independence in design by iterative redesign." *Engineering with Computers*, pages 137-145, 1987.

[East 1995] E.W. East et al. "The Reviewer's Assistant System: System Design Analysis and Description." Technical Report, U.S. Army Construction Engineering Research Laboratory, 1995.

[Echeverry 1991] D. Echeverry. "Factors for Generating Initial Construction Schedules." Technical Report, U.S. Army Construction Engineering Research Laboratory, 1991.

[Fazio 1989] P. Fazio and K. Gowri. "A knowledge-based system for the selection and design of roof systems." *The Journal of CIB Batiment International: Building Research and Practice,* 17(5)294-298, September/October 1989.

[Fischer 1991] G. Fischer et al. "Critics: an emerging approach to knowledge-based human-computer interaction." *International Journal of Man-Machine Studies,* 35(5):695-721, November 1991.

[Fischer 1993] G. Fischer et al. "Embedding critics in design environments." *The Knowledge Engineering Review,* 8(4):285-307, December 1993.

[Fu 1994] M. Fu et al. "Using a goal-based model of design in an expert critiquing system." In *Design Cognition and Design Education: Focus on the Role of Experience,* pages 14-19. EduTech Symposium, Georgia Institute of Technology, 1994.

[Goodman 1990] B. Goodman and D. Litman. "Plan recognition for intelligent interfaces." In *Proceedings of the Sixth IEEE Conference on Artificial Intelligence Applications,* 1990.

[Griffin 1982] C. Griffin. *Manual of Built-Up Roof Systems.* McGraw-Hill Book Company, New York, 1982.

[Hagglund 1993] S. Hagglund. "Introducing expert critiquing systems." *The Knowledge Engineering Review,* 8(4):281-284, December 1993.

[Ito 1989] K. Ito et al. "Linking Knowledge-Based Systems to CAD Design Data with an Object-Oriented Building Product Model." Technical Report, Center for Integrated Facility Engineering, Stanford University, 1989.

[Ito 1990] K. Ito et al. "PMAPM: An Object Oriented Project Model for A/E/C Process with Multiple Views." Technical Report, Center for Integrated Facility Engineering, Stanford University, July 1990.

[Jakiela 1988] M. Jakiela. "Intelligent Suggestive CAD Systems." PhD thesis, University of Michigan, 1988.

[Kelly 1984] V. Kelly. "The CRITTER system: automated critiquing of digital circuit designs." In *Proceedings of the 21st Design Automation Conference,* pages 419-425, 1984.

[Kowalski 1985] T. Kowalski and D. Thomas. "The VLSI design automation assistant: what's in a knowledge base." In *Proceedings of the 22nd Design Automation Conference,* pages 252-258, 1985.

[Lotvin 1984] M. Lotvin et al. "AMOEBA: a symbolic VLSI layout system." In *Proceedings of the 21st Design Automation Conference,* pages 294-300, 1984.

[Marcus 1992] S. Marcus et al. "VT: an expert elevator designer that uses knowledge-based backtracking." In *Artificial Intelligence in Engineering Design,* volume 1, chapter 11. Academic Press, Inc., New York, 1992.

[Mastaglio 1990] T. Mastaglio. "User modeling in computer-based critics." In *Proceedings of the 23rd Hawaii International Conference on System Sciences, Volume 3: Decision Support and Knowledge-Based Systems*, pages 403-411, IEEE Computer Society Press, Los Alamitos, 1990.

[Miller 1986] P. Miller. *Expert Critiquing Systems: Practice-Based Medical Consultation by Computer*. Springer, New York, 1986.

[Mittal 1986] S. Mittal and C. Dym. "PRIDE: an expert system for the design of paper handling systems." *IEEE Computer*, pages 102-114, July 1986.

[Morad 1994] A. Morad and Y. Beliveau. "Geometric-based reasoning system for project planning." *Journal of Computing in Civil Engineering*, 8(1):52-69, January 1994.

[Najem 1993] Z. Najem. "A Hierarchical Representation of Control Knowledge For A Heuristic Classification Shell," PhD thesis, University of Illinois at Urbana-Champaign, 1993.

[NRCA 1985] National Roofing Contractors Association, Chicago. *The NRCA Roofing and Waterproofing Manual*, 2nd edition, 1985.

[Ohsuga 1989] S. Ohsuga. "Toward intelligent CAD systems." *Computer Aided Design*, 21(5):315-336, June 1989.

[Ousterhout 1984] J. Ousterhout et al. "Magic: A VLSI layout system." In *Proceedings of the 21st IEEE Design Automation Conference*, pages 152-159, 1984.

[Paek 1992] Y. Paek and H. Adeli. "An object space framework for design/construction integration." *Building and Environment*, 10(1):35-48, 1992.

[Pietras 1994] C. Pietras and B. Coury. "The development of cognitive models of planning for use in the design of project management systems." *International Journal of Human-Computer Studies*, 40:5-30, January 1994.

[Pohl 1992] J. Pohl et al. "A Computer-Based Design Environment: Implemented and Planned Extensions of the ICADS Model." Technical Report, California Polytechnic State University, January 1992.

[Ramsey 1994] C. Ramsey and H. Sleeper. *Architectural Graphic Standards*. John Wiley and Sons Inc., New York, ninth edition, 1994.

[Roach 1984] J. Roach. "The rectangle placement language." In *Proceedings of the 21st IEEE Design Automation Conference*, pages 405-411, 1984.

[Roessler 1993] T. Roessler et al. "The BCOE Advisor System, Volume 1: System Design Analysis and Description." Technical Report, U.S. Army Construction Engineering Research Laboratory, Champaign, IL, 1993.

[RCI 1994] Roof Consultants Institute, Raleigh, NC. *Roof Consultants Institute's Glossary of Terms*, 1994.

[Rook 1993] F. Rook and M. Donnell. "Human cognition and the expert system interface: mental models and inference explanations." *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6):1649-1661, 1993.

[Rumbaugh 1991] J. Rumbaugh et al. *Object-Oriented Modeling and Design*. Prentice Hall, New Jersey, 1991.

[Silverman 1992a] B. Silverman. *Critiquing Human Error: A Knowledge-Based Human-Computer Collaboration Approach*. Academic Press, New York, 1992.

[Silverman 1992b] B. Silverman. "Building a better critic: recent empirical results." *IEEE Expert*, 7(2):18-24, April 1992.

[Silverman 1992c] B. Silverman. "Survey of expert critiquing systems: practical and theoretical frontiers." *Communications of the ACM*, 35(4):106-127, April 1992.

[Silverman 1992d] B. Silverman and T. Mezher. "Expert critics in engineering design: lessons learned and research needs." *AI Magazine*. 13(1):45-62, Spring 1992.

[Silverman 1993] B. Silverman and R. Wenig. "Engineering expert critics for cooperative systems." *The Knowledge Engineering Review*, 8(4):309-328, December 1993.

[Spickelmier 1988] R. Spickelmier and A. Newton. "CRITIC: a knowledge-based program for critiquing circuit designs." In *Proceedings of the 1988 IEEE International Conference of Computer Design: VLSI in Computers and Processors*, pages 324-327, 1988.

[Steinberg 1984] L. Steinberg and T. Mitchell. "A knowledge-based approach to VLSI CAD: the REDESIGN system." In *Proceedings of the 21st IEEE Design Automation Conference*, pages 412-418, 1984.

[Steinberg 1992] L. Steinberg. "Design As Top-Down Refinement Plus Constraint Propagation." In *Artificial Intelligence in Engineering Design*, volume 1, chapter 8. Academic Press, Inc., New York, 1992.

[Taylor 1984] G. Taylor and J. Ousterhout. "Magic's incremental design-rule checker." In *Proceedings of the 21st IEEE Design Automation Conference*, pages 160-165, 1984.

[Tong 1987] C. Tong. Goal-directed planning of the design process. In *Proceedings of the 1987 IEEE International Conference of Computer Design: VLSI in Computers and Processors*, pages 284-289, 1987.

[USACE 1992] U.S. Army Corps of Engineers and the American Roofing Consultants, Inc., Spencer, NC. *Roofing Technology: Proponent Sponsored Engineer Corps Training (PROSPECT)*, 1992.

[van der Meulen 1988] P. van der Meulen et al. "EXIST: an interactive VLSI architectural environment." In *Proceedings of the 1988 IEEE International Conference of Computer Design: VLSI in Computers and Processors*, pages 312-319, 1988.

[Wenger 1987] E. Wenger. *Artificial Intelligence and Tutoring Systems*. Morgan Kaufmann Publishers, Inc., Los Altos, 1987.

[Yokota 1987] T. Yokota et al. "A VLSI design automation system using frames and logic programming." In *Proceedings of the 1987 IEEE International Conference of Computer Design: VLSI in Computers and Processors*, pages 296-301, 1987.

[Zamanian 1992] M. Zamanian et al. "Representing spatial abstractions of constructed facilities." *Building and Environment*, 27(2):221-230, April 1992.

[Zhou 1989] H. Zhou et al. "CLEER: An AI system developed to assist equipment arrangements on warships." *Naval Engineers Journal*, 101(3):12-137, 1989.

# Acronyms and Abbreviations

A/E/C        Architect/Engineer/Contractor

CAD          computer-aided design

CMA          critic management agent

DCCA         design code critic agents

DTM          Designer's Task Model

EPDM         ethylene propylene diene monomer

HVAC         heating, ventilating, and air conditioning

ICAI         intelligent computer-assisted instruction (system)

IDT          Intelligent Design Tool

ITS          intelligent tutoring system

NRCA         National Roofing Contractors Association

RCI          Roof Consultants Institute

SEDAR        Support Environment for Design And Review

USACERL  U.S. Army Construction Engineering Research Laboratories

## USACERL DISTRIBUTION

Chief of Engineers
    ATTN: CEHEC-IM-LH (2)
    ATTN: CEHEC-IM-LP (2)
    ATTN: CECC-R
    ATTN: CEMP-C
    ATTN: CEMP-CE (2)
    ATTN: CEMP-E
    ATTN: CEMP-ES (2)
    ATTN: CERD-L

US Army Engr District
    ATTN: Library (40)
    ATTN: Civil Engineers (40)

US Army Engr Division
    ATTN: Library (11)
    ATTN: Civil Engineers (11)
    ATTN: Civil Construction/Civil Con-Ops (11)

Defense Tech Info Center 22060-6218
    ATTN: DTIC-O (2)

                                        127
                                        7/96